The Pragmatic Programmers

TURING 图灵程序设计丛书

通过解决问题锤炼软件开发技能

挑战编程技能

57道程序员功力测试题

[美] Brian P. Hogan 著

臧秀涛 译

中国工信出版集团 人民邮电出版社 POSTS & TELECOM PRESS

数字版权声明

图灵社区的电子书没有采用专有客户端，您可以在任意设备上，用自己喜欢的浏览器和PDF阅读器进行阅读。

但您购买的电子书仅供您个人使用，未经授权，不得进行传播。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。

Brian P. Hogan

Web开发者、教师、作者、编辑。自1995年起一直在开发Web站点和应用。曾是多家小企业的自由开发者，并在两家创业公司担任过技术主管。经常在各种技术大会上发表演讲。目前在契皮瓦谷技术学院教授软件开发课程。喜欢撰写技术著作，尤其是关于Web设计和开发的。另著有《HTML5和CSS3实例教程》《Web开发秘方》等书。Twitter账号@bphogan。



臧秀涛

中国科学院计算技术研究所硕士，对编程语言、虚拟机等领域有浓厚兴趣。先后从事过游戏服务器、操作系统方面的开发，现于InfoQ任QCon大会主编，推动软件开发领域内实践经验的传播。业余喜欢读书、翻译，译作包括《Java性能权威指南》《C++ Primer中文版》《C++ API设计》《Groovy程序设计》等。期待读者通过微博（@臧秀涛）或者微信公众号dev-news联系。

TURING 图灵程序设计丛书

挑战编程技能

57道程序员功力测试题

[美] Brian P. Hogan 著
臧秀涛 译

人民邮电出版社
北京

图书在版编目（C I P）数据

挑战编程技能 ： 57道程序员功力测试题 / （美） 布莱恩·霍根（Brian P. Hogan）著 ； 臧秀涛译. -- 北京: 人民邮电出版社，2017.2
（图灵程序设计丛书）
ISBN 978-7-115-44680-0

Ⅰ. ①挑… Ⅱ. ①布… ②臧… Ⅲ. ①程序设计－习题集 Ⅳ. ①TP311.1-44

中国版本图书馆CIP数据核字(2017)第009877号

内 容 提 要

新手程序员在具备了理论基础后，面对实际项目时往往不知道如何解决问题；有经验的程序员在学习了一门新语言后，也会有很多不知道如何使用的特性。针对程序员的这一普遍困惑，资深软件工程师 Brian P. Hogan 在这本书中总结了 57 道练习题，帮助他们锤炼技能。这些练习题均取自实践，难度会逐渐增加，使得编程训练充满挑战又乐趣多多。

本书适合所有程序员阅读。

◆ 著　[美] Brian P. Hogan
译　臧秀涛
责任编辑　岳新欣
责任印制　彭志环

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷

◆ 开本：880×1230　1/32
印张：4.25
字数：106千字　2017年2月第1版
印数：1－3 500册　2017年2月北京第1次印刷

著作权合同登记号　图字：01-2015-8282号

定价：39.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京东工商广字第 8052 号

版权声明

如何使用本书

宝剑锋从磨砺出。

音乐会钢琴家每天练习数小时，学习音乐，排练技艺，提升技能。他会反复练习同一段音乐，打磨每一处小细节，确保准确无误，因为等到登台表演时，面对那些花钱、花时间来的观众，他希望可以为自己的表现而感到骄傲。

职业足球运动员会在健身房里花费很多时间，练习托举、跑动和跳跃，并反复训练，直至完全掌握，然后才开始练习足球。他会研究比赛，观看以前的比赛录像。当然，他也会参加分组对抗和表演赛，确保做好一切准备，可以随时真正上场比赛。

空手道从业者一生都在练习“形”（kata）——这是模仿实战的动作套路，学习如何呼吸以及正确发力。他会千万遍地练习同一个套路，通过一次次重复做到越来越好。

我遇到过的优秀软件开发者，也是这样锤炼其技艺的。他们并不是天天只利用上班时间练习，而是会用自己的时间学习新编程语言，在其他方面不断精进自己的技术。当然，在工作中他们也会学习新东西，

但因为是工作时间，所以会有预期：公司是希望你有产出的，而不是花钱请你来练习的。

本书要谈的就是程序员如何锤炼其技能。翻到本书的某一页，打开文本编辑器，敲出上面的程序。可以自己做些修改。用你从来没用过的一门语言实现这个程序。随着一次次重复，你要做得越来越好。

目标读者

本书主要面向两类程序员。

首先，本书为刚开始学习编程的人提供了课外练习材料。技能不可能仅靠做作业就得到提升。未来的雇主会希望你们表现出批判性思维和解决问题的能力，而这需要通过练习来训练。本书会提供这种练习，书中的问题都取自实践，是很多开发者都要面对的，但是这里针对读者的能力做了针对性的设计。每一章都会围绕一项基本的编程能力展开，而且会比前一章复杂一点，后面的内容构建于前面所学的知识之上，同时让你为以后的挑战做好准备——不管是课内的还是课外的。

很多新手程序员习惯于有人清清楚楚地告诉他们如何解决某个问题。他们往往是跟着可以直接输入示例代码的教程来学习一门语言。这确实也是开始编写代码的不错途径。但是当面对没有现成答案的开放性问题时，这些程序员就纠结了。任何有经验的人都可以告诉你，软件开发中满是开放性问题。本书中的练习可以帮助你培养问题求解技能，使你有信心攻克更大的问题——或许是从来没有人解决过的问题。

不过，本书也适合那些想将手头的事做得更好的有经验的程序员。在学习 Go 和 Elixir 时，我使用了类似本书中的那些程序。在尝试 iOS 开发时，我也编写了这些程序。每隔一段时间，我都会用一门我了解的语言实现一下它们。我熟练掌握了 JavaScript 和 Ruby，而看看是否能以不同的方式、不同的算法或模式实现其中的一个程序，也是极大的挑战。在我开始全职讲授 Ruby 和 JavaScript 时，这些程序帮我发现和解释了这两种语言的一些特性，这些特性我知道怎么用，但是并没有充分理解。所以，如果你是有经验的开发者，我鼓励你也这么做。比如用 Haskell 试试其中的一个程序；或是试试用你熟悉的每种语言编写其中的一个，然后比较一下结果。也可以让你的同事每周做其中的一个练习，并比较你们的方案；或是使用这些程序来指导团队中的新人。

教育工作者需要注意

如果你是在高中或大学讲授编程入门课程，可能会发现本书中的练习在课堂上也很有用。不过，我不建议将其用作最终的考试评估，我鼓励读者彼此分享其解决方案。当然，如果学生可以共同合作，我也建议将本书中的练习用于课堂。这些练习很适合问题导向的学习环境。

本书内容

之所以写作本书，首先是希望向编程语言初学者提供一些一开始可能要面对的、有挑战的问题。因此，大部分问题一开始相对简单，难度会逐渐增加。书中练习的这种延续性，使得编程基础的训练充满挑战，

同时又有很多乐趣，还可以让我们更加快速地上手一门新语言。在第一部分，程序只是简单地接受一些输入，操作数据得到不同的输出，让你体验计算机程序如何处理输入和输出操作。这是新手第一周要编写的程序。

下一步就有些挑战了，要编写进行计算的程序。有些程序就像计算房屋面积那么简单，但是其他程序可能会涉及金融和医疗方面的计算，和我们可能在工作中碰到的类似。

然后，通过引入决策逻辑和重复逻辑，我们会增加程序的复杂性，而且会引入函数。

之后，你会发现有些问题需要使用像数组（array）或映射（map）这样的数据结构来解决。这些程序还需要你利用之前解决的其他问题。

当然，没有点儿文件输入输出，程序是不完整的。所以我们会练习从文件中读数据，处理，最后写回。

现代的程序往往要和外部服务交互，因此你会发现一些程序需要使用第三方 API 的数据。

最后，本书后面有几个比较大的程序，需要读者综合前面所学。

此外，每个练习都有一些在构建程序时必须遵循的约束，还有一些高于当前程序的挑战。如果你完全没有编程基础，可以跳过这些挑战，等技能提高之后再回来看看。如果你已经有些经验，并且感觉程序太过简单，则可以立刻试试。取决于所选择的编程语言，有些挑战可能非常困难。比如，如果使用 JavaScript 和 HTML 开发这些程序，那实现一个 GUI 版本会非常容易。如果使用 Java 开发，就要多做大量的

工作。不过对于这些挑战，如果你感觉合适，可以自由修改。

然而，本书不会提供这些程序的解决方案。如果你做了足够深入的思考，而且使用了所有可以调配的资源，应该能独立想出解决之道，而这正是本书主旨所在。

最后一件事：本书中没有那些低劣的面试题，也没有 FizzBuzz 之类的问题。你不需要翻转二叉树，也不需要编写快速排序算法（除非它作为某种解决方案的一部分而用到）。如果你想找这类内容，应该到别的地方看看。那类问题有价值，但至于为什么要做，我们并不清楚，所以实际做起来会更困难。这就使得人们感觉它们不是那么好接近，因而成了学习的障碍。

本书中的问题是比较简单而且容易涉及的现实问题，可以帮你练习使用代码解决问题。

读者所需

你需要的就是自己喜爱的开发环境，甚至可以是从来没用过的那种。本书没有限定于某种具体的编程语言。请选择一门语言，抓起这门语言的参考指南，深入进去。但是提醒一句，编程语言的选择会影响书中程序实现的难易。比如，如果选择 Python 或 Ruby 来配合学习本书，开发图形用户界面就没那么容易。如果选择在浏览器中使用 JavaScript，那么使用外部文件和 Web 服务就会比其他语言复杂。与面向对象语言相比，如果选择的是函数式语言，则问题的解决方案会有显著区别。不过这正是这些练习的真正价值所在；它们可以帮你学习一门语言，并理解它与你之前学过的语言有何差别。

你应该能接入互联网，这样就可以尝试使用第三方服务的程序，并参与到本书的社区中来。

在线资源

本书的配套网站[1]有一个论坛，你可以在那里和其他开发者探讨本书。你可以随意用自己钟爱的编程语言提交解决方案，并和其他读者探讨自己的方案。人们可以以不同方式解决问题，每个开发者都可以有自己的风格，这是编程最迷人的地方之一。

电子书

扫描如下二维码，即可购买本书电子版。

① http://pragprog.com/titles/bhwb

致 谢

首先，感谢读者选择这本书，并愿意提升自己的软件开发技能。你真棒。这本书就是为像你这样的读者编写的，感谢阅读。

其次，感谢 Dave Thomas。感谢你相信本书的理念，也感谢你这些年来对我的指导。能向你学习是一种荣耀，我深感荣幸。在本书的写作上，你对我的鼓励意义非凡，感谢你投入很多时间审核书中的习题并提出建议。你和 Andy 让程序员的世界越来越好，非常感谢你们让我有机会以自己的方式作出一点小贡献。

特别感谢 Susannah Pfalzer。你让我的书变得更好。你似乎总能准确地抓住所有细节，而且指导我专注在真正重要的地方。这是你帮我做的第 6 本书，因为你这些年的指导，我才成了更好的作者。

还要感谢 Andy Hunt、Mike Reilly、Michael Swaine、Fahmida Rashid 和 Bruce Tate，感谢你们在我提出这些理念时给予我的鼓励。

本书中的程序都是我过去 10 年在编程教学中用过的。感谢 Zachary Baxter、Jordan Berg、Luke Chase、Dee Dee Dale、Jacob Donahoe、Alex Eckblad、Arrio Farugie、Emily Mikl、Aaron Miller、Eric Mohr、

Zachary Solofra、Darren Sopiarz、Ashley Stevens、Miah Thalacker、Andrew Walley，以及这些年上过我的课或者参加过我的培训的其他所有学生。你们对教学方法的反馈让我获益良多。感谢 Kyle Loewenhagen、Jon Cooley 和 George Andrews，你们的反馈和意见帮助我成长为一名更优秀的教师。

感谢 Deb Walsh，感谢你的鼓励，以及关于如何让学生学得更好的奇思妙想。关于教学，我们有共同的核心理念，在和你的交流中我也学到很多。感谢你与我分享自己的经验和专业知识，也感谢你对我的教学方法的支持。

本书凝结了很多新老开发者的很棒的反馈，这使得本书的习题衔接得更为合理、顺畅。出版前期审校人员花了很多时间和精力，用他们喜欢的编程语言实现书中的习题，并帮我找出了一些没有意义或需要改进的地方。感谢 Chris C.、Alex Henry、Jessica Janiuk、Chris Johnson、Aaron Kalair、Sean Lindsay、Matthew Oldham、Stephen Orr、Jason Pike、Jessica Stodola、Andrew Vahey 和 Mitchell Volk，感谢你们付出宝贵的时间测试这些习题，并提供建议和反馈。

感谢我的合作伙伴 Mitch Bullard、Kevin Gisi、Chris Johnson、Jeff Holland、Erich Tesky、Myles Steinhauser、Chris Warren 和 Mike Weber 对我的支持。

感谢我的妻子、最好的朋友 Carissa。你的爱和支持使本书成为可能。我会永远感激你为我以及女儿们所做的一切。

最后，感谢棒棒的 Ana 和 Lisa，感谢你们在我写作时给我拥抱，给我发信息。感谢你们在我写下这段文字时陪伴在我身旁。

目　　录

第 1 章

将问题转变成代码

如果你才接触编程，可能想知道有经验的开发者是如何看问题以及如何将其转变为可运行的代码的。其实，编写代码只是这个过程中很小的一部分。在解决问题之前，首先要将其分解。如果你观察过有经验的开发者，可能会觉得他们看上去就是打开代码编辑器，噼里啪啦敲出了一个解决方案。但是多年以来，他们分解的问题就算没有几千个，也有几百个了，他们能看出一些模式。而新手可能并不知道怎么做。所以，本章将研究一种分解问题并将其转变为代码的方法。你可以使用该方法解决本书其余部分的问题。

理解问题

要想知道应该做什么，最好的一个方法是把它写下来。如果我告诉你，我想要一个小费计算程序，是不是凭这个信息你就能开工写一个了？可能还不够。你可能还必须向我提一些问题。这通常称作收集需求，不过我喜欢把它看作找出程序应该提供的特性。

先考虑一些可以向我提出的问题，以便通过这些问题更好地理解我想要的是什么。要构建这个应用，你需要了解什么呢？

想到一些问题了吗？很好。下面是一些你可能想问的问题。

- 你想使用什么样的公式？可以解释一下如何计算小费吗？
- 小费的比例是多少？是固定的 15%，还是用户可以修改？
- 程序启动后，屏幕上应该显示什么？
- 程序应该如何显示其输出？你想看小费和总额，还是只想看总额？

一旦得到这些问题的答案，就可以尝试写一下问题描述，精确地解释要构建的东西。下面是欲构建程序的问题描述：

> 创建一个简单的小费计算程序。该程序应该提示用户输入账单金额和小费比例。该程序必须计算出小费，并显示小费和总金额。

示例输出：

```
What is the bill? $200
What is the tip percentage? 15
The tip is $30.00
The total is $230.00
```

> **小乔爱问：**
> **如何处理复杂的程序？**
>
> 将较大的程序分解成更容易管理的、较小的特性。这么做的话，因为每个特性可以具体化，所以成功的概率更大。已有的大部分复杂应用都是由很多较小的、一起工作的程序组成的。Linux 上的命令行工具就是这么工作的；一个程序的输出可以作为另一个程序的输入。

如果你这就准备打开文本编辑器敲代码，还是有点太着急了。听我说，

如果不花时间仔细设计，或许也能开发出能够工作的程序，但是质量并不高。而且不幸的是，很容易出现一些不好的事情。比如，你写出的程序没有测试、计划，也没有文档，而你的老板看到这个程序后，以为完事了，让你发布。现在产品中有未经测试、没有计划的代码，以后可能还会有人让你做些修改。设计差劲的代码很难维护和扩展。所以就让我们以小费计算程序为例，从头到尾过一遍这个能够帮你理解应该构建什么的简单过程。

发现输入、处理和输出

不管是小费计算程序这样的简单应用，还是 Facebook 那样的复杂应用，每个程序都有输入、处理和输出。事实上，大型应用就是由一系列较小的、相互通信的程序组成的。其中，一个程序的输出会成为另一个程序的输入。

不管程序是大是小，如果能花点时间清晰地阐明输入、处理和输出各是什么，就能确保其工作地很好。如果有比较清晰的问题描述，则有个简单的方法，就是看一下问题描述中的名词和动词。名词最终会成为输入和输出，动词会成为处理。看一下小费计算程序的问题描述：

> 创建一个简单的小费计算程序。该程序应该提示用户输入账单金额和小费比例。该程序必须计算出小费，并显示小费和总金额。

首先，找找名词。如果你喜欢，可以拿笔圈起来，也可以列个表。下面是我列的表：

- 账单金额
- 小费比例

- 小费
- 总金额

再来看看动词有哪些呢？

- 提示
- 计算
- 显示

所以我们知道了，必须提示用户输入，做些计算，然后显示输出。通过考虑名词和动词，可以了解要求我们做的是什么。

当然，问题描述未必总是这么清晰。比如，问题描述说需要计算小费，但又说需要显示小费和总金额。这意味着需要把小费和原始的账单金额相加，得到那个输出。这是构建软件的挑战之一。它未必总会被讲清楚。但随着经验越来越丰富，你将能弥合这种差异，领会字里行间的言外之意。

借助一点侦查工作，我们确定了这个程序的输入、处理和输出，如下所示。

- 输入：账单金额，小费比例
- 处理：计算小费
- 输出：小费、总金额

我们是不是已经为编写代码做好准备了呢？还没有。

用测试驱动设计

设计和开发软件的最佳方式之一，是从一开始就思考想要的结果。很

多职业软件开发人员会采用一种叫作测试驱动开发（Test-Driven Development，TDD）的正规过程做到这一点。在 TDD 中，你会编写很多代码测试程序的输出，或者是测试组成大型程序的单个程序的输出。随着开发的进行，这个测试过程会指导你走向好的设计，并帮助你思考程序可能存在的问题。

TDD 确实需要一些所用语言相关的知识，还需要一点初学者尚不具备的经验。

然而，TDD 的本质是提前思考程序的预期结果是什么，然后为此努力。如果在编写代码之前做到这一点，我们的思维就能超越最初的需求。即使我们并不能很舒服地按照 TDD 的正规流程做下来，只是创建一些简单的测试计划，也有不少好处。测试计划会列出程序的输入和预期结果。

测试计划看上去就像这样：

```
输入：
预期结果：
实际结果：
```

列出程序的输入，然后写出应该得到的输出。运行程序，比较预期结果与程序给出的实际结果。

让我们通过思考小费计算程序来实践一下。我们怎么知道程序的输出应该是什么呢？我们怎么知道计算是否正确呢？

通过一些测试计划来定义我们希望程序如何工作。先写一个非常简单的：

```
输入：
  账单金额：10
```

```
  小费比例：15
预期结果：
  小费：$1.50
  总金额：$11.50
```

这个测试计划告诉我们两件事。首先，它告诉我们，程序需要两个输入：账单金额 10 和小费比例 15。因此，在做数学运算时，需要将小费比例从整数转换成小数。它还告诉我们，我们将以货币形式打印小费和总额。所以我们知道了，程序中最好做些转换。

一个测试并不够。如果使用 11.25 作为输入呢？测试计划中，输出应该是什么？试试看。填充下面的测试计划：

```
输入：
  账单金额：11.25
  小费比例：15
预期结果：
  小费：???
  总金额：???
```

假设你用计算器算出了小费，得到的结果或许会是 1.6875。

但是这现实吗？可能并不现实。我们可能会向上取整到最接近的分上。所以测试计划将是这样的：

```
输入：
  账单金额：11.25
  小费比例：15
预期结果：
  小费：$1.69
  总金额：$12.94
```

我们刚刚使用一个测试来设计了程序的功能；我们确定了程序需要对答案中的分做向上取整。

在你做本书中的练习时，花些时间，为每个程序设计至少 4 个测试计划，并尝试尽可能多地考虑别人可能会使程序出错的场景。随着接触更复杂的问题，你可能需要更多的测试计划。

如果你是想从 TDD 开始的有经验的开发人员，则应该使用本书中的练习来熟悉你喜爱的编程语言所提供的库和工具。可以在维基百科[①]上找到很多编程语言中有的测试框架。可以阅读 Kent Beck 的《测试驱动开发》来获得更多信息，了解如何设计带测试的代码，还可以调研更多与 TDD 相关的更特定于语言的资源。

现在我们对小费计算程序应该具有的特性有了更清晰的认识，可以开始组合程序的算法了。

用伪代码编写算法

算法就是需要执行的一组一步一步的操作。如果拿到一个算法，编写代码执行这些操作，最后会得到一个计算机程序。

如果才学编程，尚未完全熟悉一门编程语言的语法，则应该考虑用伪代码把算法写出来。伪代码语法和英语类似，方便我们思考逻辑，还不用担心纸的问题。伪代码并不是只有初学者才用得着；在和同事一起解决问题时，甚至是一个人解决问题时，有经验的程序员偶尔也会在白板上写一些伪代码。

编写伪代码并没有所谓“正确的方式”，尽管有些应用较广的术语。可以使用 `Initialize` 表示设置初始值，使用 `Prompt` 表示提示用户

① https://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

输入，以及使用 Display 表示将内容显示到屏幕上。

小费计算程序用伪代码可以这样写：

```
TipCalculator
  Initialize billAmount to 0
  Initialize tip to 0
  Initialize tipRate to 0
  Initialize total to 0

  Prompt for billAmount with "What is the bill amount?"
  Prompt for tipRate with "What is the tip rate?"

  convert billAmount to a number
  convert tipRate to a number

  tip = billAmount * (tipRate / 100)
  round tip up to nearest cent
  total = billAmount + tip

  Display "Tip: $" + tip
  Display "Total: $" + total
End
```

从这段伪代码中可以大致看出程序的算法。必须设置一些变量，根据输入作出决策，进行转换，然后将输出显示到屏幕上。我建议在伪代码中，包含像变量的名字和要在屏幕上显示的文本等细节，因为这可以帮助我们更清晰地思考程序的最终结果。

这是编写程序的最佳方式吗？或许不是。但这并不是关键所在。通过编写伪代码，我们创建出了一些可以演示给其他开发者以获得反馈的东西。把它们扔到一块花不了太多时间。

最好的情况是，我们可以将其用作一个蓝图，可以用任何编程语言编码实现。请注意，伪代码中并没有假定最终会使用何种编程语言，但

是确实提供了一些指导，像变量名是什么，显示给最终用户的输出是什么，等等。

一旦有了程序的一个初始版本，并让它跑起来，就可以开始修改代码来加以改进了。比如，可能是将程序划分成函数，或者是将数学转换放到一起做，而不是分开进行。就是把伪代码看作一种计划工具。

编写代码

现在轮到你了。利用所学的知识，是不是可以编写出这个程序的代码了？试试吧。在做的时候要记住下面这些约束。

约束

- 小费比例应输入百分比的数字部分。比如，15%的小费比例，应该输入 15，而不是 0.15。应该由程序来处理除法。
- 不足一分的，向上取整。

如果不知道如何实施这些约束，可以暂不考虑，先编写程序，以后再回来看。这些习题的目的就是练习和提高。

如果现在感觉这个程序挑战很大，可以先跳过，先做做本书中那些更容易的，然后再回到这个问题。

挑战

在编写出一个基本的版本后，可以再试试解决下面这些挑战。

- 对于账单金额和小费比例，确保用户只能输入数值。如果用户输入的不是数值，则显示相应的提示信息并退出程序。下面是测试计划的一个例子：

```
输入：
    账单金额：abcd
    小费比例：15
预期结果：请输入一个合法的账单金额数值。
```

- 不再是显示错误消息并退出，而是在用户输入合法的值之前，一直提示输入。
- 不允许用户输入负数。
- 将程序分解为做计算的函数。
- 使用图形用户界面（GUI）实现这个程序，当有任何值改变时，自动在界面中更新这些值。
- 不再是让用户以百分比形式输入小费比例，而是让用户通过拖动一个滑动条表示对服务员的满意度，区间是 5%~20%。

前进！

尝试使用该策略解决本书中的每个问题，从而使收益最大化。发现输入、处理和输出。开发些测试计划，想出伪代码，编写程序，然后接受每个程序后面的各种挑战。或者是选择自己的方向。再或者是用尽可能多的语言编写这个程序。

但最重要的是，玩得开心，享受学习。

第 2 章

输入、处理和输出

从用户那里获取输入，并将其转换为有意义的东西，这是编程中很基础的一个部分。软件开发人员总是将数据变为可用于支持决策的信息。这些数据可能来自键盘、鼠标、触摸板、刷卡机，甚至游戏控制器。计算机必须作出响应，加以处理，并做些有意义的事。

本章的练习题将帮助你了解如何从用户那里获取输入，以及如何处理输入以得到输出。你将构建字符串，做些数学运算，并感受一下所用的编程语言。这些问题很简单，但是将帮助你建立作为程序员的自信心；后面章节中的问题会更复杂。

每个练习都设有一些挑战，如果力所能及，可以试试。如果是刚学编程，有些挑战需要用到一些你可能尚不熟悉的技术，你可以随意跳过；你总是可以稍后再回来接受这些挑战。

准备好了吗？出发！

1 问好

在很多编程语言中，“Hello, World”程序都是你要学习编写的第一个程序，但是它没有涉及任何输入。

所以先来创建一个提示输入名字并打印包含该名字的问候信息的程序。

示例输出

```
What is your name? Brian
Hello, Brian, nice to meet you!
```

约束

- 输入、字符串连接和输出这几个部分要分开。

挑战

- 不使用任何变量，编写一个新版本的程序。
- 编写一个新版本，对不同的人显示不同的问候语。在完成第 4 章和第 7 章的练习后，这是个值得一试的不错的挑战。

2　计算字符数

创建一个程序，提示用户输入字符串，然后输出这个字符串，以及其中包含的字符数。

示例输出

```
What is the input string? Homer
Homer has 5 characters.
```

约束

- 确保输出中包含原始的字符串。
- 使用一个输出语句来构造输出。
- 使用所用编程语言中的一个内置函数来确定字符串的长度。

挑战

- 如果用户什么都没输入，提示用户输入内容。
- 使用图形用户界面实现该程序，在每次键盘按下时，更新字符计数信息。如果所用语言没有特别友好的 GUI 库，尝试使用 HTML 和 JavaScript 来做这个练习。

3 打印引语

引号常用于表示一个字符串的开始和结束。但有时我们需要使用转义字符打印引号本身。

创建一个程序，提示用户输入一条引语及其作者。按照示例输出那样显示引语和作者。

示例输出

```
What is the quote? These aren't the droids you're looking for.
Who said it? Obi-Wan Kenobi
Obi-Wan Kenobi says, "These aren't the droids
you're looking for."
```

约束

- 使用一条输出语句来生成该输出，使用适当的字符串转义技术来处理引语。
- 如果所用编程语言支持字符串插入或替换，在这个练习中不要使用。要使用字符串连接。

挑战

- 在第 7 章中，你将练习使用数据的列表。修改这个程序，不再提示用户输入引语，而是自己创建一个保存引语及其作者的结构，然后使用例子中的格式显示所有的引语。由映射（map）组成的数组会是个不错的选择。

4 疯狂填词

疯狂填词是一个简单的游戏，我们创建一个空出一些单词的故事模板。你或另一个玩家提供一系列单词，将其放到故事中，得到一个通常或愚蠢或搞笑的故事。

创建一个简单的疯狂填词程序，提示用户输入一个名词、一个动词、一个形容词和一个副词，然后将这些词插入到所创建的故事中。

示例输出

```
Enter a noun: dog
Enter a verb: walk
Enter an adjective: blue
Enter an adverb: quickly
Do you walk your blue dog quickly? That's hilarious!
```

约束

- 在程序中使用一条输出语句。
- 如果所用语言支持字符串插入或替换，则使用它来构造输出。

挑战

- 向程序中添加更多输入，扩展故事。
- 实现一个带有分支发展的故事，可以根据问题的答案来确定如何构造故事。在第 4 章的问题中，你将进一步探索这个概念。

5 简单的数学处理

你将经常编写处理数值的程序。取决于所使用的编程语言，有时必须将获取的输入转换成数值数据类型。

编写一个程序，提示用户输入两个数。打印这两个数的和、差、积及商，如示例输出所示。

示例输出

```
What is the first number? 10
What is the second number? 5
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2
```

约束

- ❑ 用户输入的值将会是字符串。确保在做数学运算之前将其转换成数值。
- ❑ 将输入和输出与数值转换及其他处理分开。
- ❑ 生成一条在适当的位置包含换行的输出语句。

挑战

- ❑ 修订这个程序，确保输入以数值形式提供。如否，不允许用户继续。
- ❑ 不允许用户输入负数。
- ❑ 将程序分解成处理计算的函数。你将在第 5 章中探索函数。
- ❑ 将该程序实现为图形用户界面程序，当任何值发生变化时，自动更新相关的值。

6　计算退休时间

你的计算机知道今年的年份，这意味着可以将该信息加到你的程序中。你只需要知道所用编程语言如何提供该信息。

创建一个程序，确定用户还有多少年退休，并计算退休年份。它应该提示用户输入当前年龄，以及想要退休的年龄，最终按如下例子这样显示输出。

示例输出

```
What is your current age? 25
At what age would you like to retire? 65
You have 40 years left until you can retire.
It's 2015, so you can retire in 2055.
```

约束

- 再次强调，在做任何数学运算之前，务必将输入转换成数值数据。
- 不要将当前的年份硬编码到程序中。通过所用编程语言从系统时间中获取。

挑战

- 处理程序返回负数的情况，提示用户已经可以退休了。

本章回顾

这些问题相当简单，但希望它们能让你思考保持输入、处理和输出分

开这一点。当程序很简单时，在其输出语句中做点数学运算或字符串连接，可能很有诱惑力，但是随着程序愈加复杂，你会发现需要将其分解为可以复用的组件。你会为一开始的这种训练而感到开心的。

赶紧开始下一章吧，是时候看些更严肃的数学问题了。

第 3 章

计　　算

你已经做过一些基本的数学运算，是时候深入研究点复杂的运算了。本章的练习有点挑战。你会用到数值转换公式，也会开发一些真正的金融相关程序。

这些程序会考查运算优先级知识。加、减、乘、除、幂和括号的优先级从高到低依次为：

- 括号
- 幂
- 乘
- 除
- 加
- 减

就算你不希望如此，计算机也总是会遵循这些规则。本章的习题会让你思考在程序中加上括号，以保证输出的正确性。

因为要处理精度问题，所以这里也需要好好利用测试计划。在很多编程语言中，如果使用小数，可能会碰到一些有趣但出人意料的结果。

比如，在 Ruby 中将 0.1 和 0.2 相加，会得到如下结果：

```
> 0.1 + 0.2
 => 0.30000000000000004
```

在 JavaScript 中也是如此。乘法更有趣。看一下如下代码：

```
> 1.25 * 0.055
 => 0.06875
```

答案是不是应该舍入到 0.06，或是四舍五入到 0.07？这完全取决于你的业务逻辑。如果答案必须是整数，可能要向上取整。

一旦涉及货币，情况会更糟。对初学编程的人而言，在尝试使用浮点数表示货币时，有个最常见的问题：浮点数会导致精度错误。

常用的一种方案是使用整数表示钱。不用 1.25，而用 125。然后进行数学运算，算完之后再把小数点放回去。下面是一个例子，还是用 Ruby：

```
> cents = 1.25 * 100.0
 => 125.0
> tax = cents * 0.055
 => 6.875
> tax = tax.round / 100.0
 => 0.07
```

你可能需要更精确。浮点数的精度问题在很多编程语言中都存在，所以有一些方便处理货币的库。比如，Java 提供的 `BigDecimal` 数据类型，甚至支持我们指定所需的舍入方式。在面对这类问题时，请仔细考虑如何处理精度。在解决现实问题，尤其是某类金融问题时，要了解舍入相关的业务逻辑。

在深入之前，还有一件事要注意：对于有编程经验的读者来说，看到最后，可能会感觉本章的练习有些重复。但是对于初学者而言，重复可以快速建立自信。在体育运动中不断演练，在学习音乐时反复训练音阶，也是此理。通过解决一些类似的问题，你可以培养解决问题的技能，同时提高分解问题的速度。而这会让你在工作上取得成功。

7 矩形房间的面积

如果你在一个全球化的环境中工作，则必须同时提供公制和英制单位的信息，而且需要知道何时进行转换，以保证结果最为精确。

编写一个计算房间面积的程序。提示用户输入以英尺为单位的房间的长和宽。然后显示以平方英尺和平方米为单位的面积。

示例输出

```
What is the length of the room in feet? 15
What is the width of the room in feet? 20
You entered dimensions of 15 feet by 20 feet.
The area is
300 square feet
27.871 square meters
```

转换公式为：

$$m^2 = f^2 \times 0.09290304$$

约束

- 让计算与输出分离。
- 使用一个常量来保存转换因子。

挑战

- 修改程序，确保输入的是数值。如果不是，不允许继续。
- 开发一个新版本，支持用户选择输入的单位是英尺还是米。
- 以 GUI 方式实现该程序，当任何一个值有修改时，自动更新结果。

8 比萨聚会

除法并不总是精确，有时候需要写程序来处理余数，而不是用小数。

编写一个平均分配比萨的程序。提示用户输入就餐人数、比萨数，以及每个比萨分几块。确保平均分配。显示每个人能得到几块。如果有剩余，显示剩下几块。

示例输出

```
How many people? 8
How many pizzas do you have? 2

8 people with 2 pizzas
Each person gets 2 pieces of pizza.
There are 0 leftover pieces.
```

挑战

- 修改程序，确保输入的是数值。如果不是，不允许继续。
- 修改输出，使其可以正确处理复数，比如：

  ```
  Each person gets 2 pieces of pizza.
  ```

  或

  ```
  Each person gets 1 piece of pizza.
  ```

  对于剩下的块数，也这样处理。

- 开发一个该程序的变种，提示用户输入就餐人数和每个人想要的比萨块数，计算需要购买多少个比萨。

9 涂料计算程序

有时必须向上取整到下一个数，而不是遵循标准的舍入规则。

计算粉刷天花板需要多少加仑涂料。提示输入长和宽，假设 1 加仑可以粉刷 350 平方英尺。以整数形式显示粉刷该房间的天花板所需的加仑数。

示例输出

```
You will need to purchase 2 gallons of
paint to cover 360 square feet.
```

记住，涂料必须整加仑购买。所以要向上取整到下一个整数。

约束

- 使用一个常量来保存转换率。
- 确保向上取整到下一个整数。

挑战

- 修改程序，确保输入的是数值。如果不是，不允许继续。
- 实现对圆形房间的支持。
- 实现对 L 型房间的支持。
- 实现一个移动版本的程序，这样在五金店也能使用。

10　自助结账

在处理多个输入和货币时，可能会引入一些微妙的精度问题。

创建一个简单的自助结账系统。提示输入 3 种商品各自的价格和数量。计算税前总价。然后以 5.5%的税率计算税额。打印商品的数量和总价，然后打印税前总价、税额和总价。

示例输出

```
Enter the price of item 1: 25
Enter the quantity of item 1: 2
Enter the price of item 2: 10
Enter the quantity of item 2: 1
Enter the price of item 3: 4
Enter the quantity of item 3: 1
Subtotal: $64.00
Tax: $3.52
Total: $67.52
```

约束

- 将程序的输入、处理和输出各部分分开。收集输入，进行数学运算并构建字符串，然后打印输出。
- 在进行任何计算之前，务必显式地将输入转换为数值数据。

挑战

- 修改程序，确保价格和数量都是以数值形式输入的。如否，则不允许继续。
- 修改程序，不再限定为仅支持 3 种商品。当没有更多商品时，计算税额和总价。

11 货币兑换

某个时候，你可能要处理货币汇率，计算要尽量精确。

编写一个货币兑换程序。具体而言，是将欧元兑换成美元。提示输入手中的欧元数，以及欧元的当前汇率。打印可以兑换的美元数。货币兑换的公式为：

$$amount_{to} = \frac{amount_{from} \times rate_{from}}{rate_{to}}$$

其中

- $amount_{to}$ 是美元数
- $amount_{from}$ 是欧元数
- $rate_{from}$ 是欧元的当前汇率
- $rate_{to}$ 是美元的当前汇率

示例输出

```
How many euros are you exchanging? 81
What is the exchange rate? 137.51
81 euros at an exchange rate of 137.51 is
111.38 U.S. dollars.
```

约束

- 注意小数部分，不足 1 美分的向上取整。
- 使用单条输出语句。

挑战

- 构建一个汇率字典，提示用户输入国家而不是汇率。
- 将应用连接到一个提供当前汇率的外部 API[①]。

① https://openexchangerates.org/是个不错的例子。

12 计算单利

计算单利是快速确定投资是否有价值的不错方式。它也能让你熟悉如何在程序中显式控制运算顺序。

编写一个计算单利的程序。提示输入本金、利率及时间，显示到期总额（本金+利息）。

单利的公式是 $A = P(1 + rt)$，其中 P 为本金，r 为年利率，t 为投资年限，A 是这笔投资到期后的总额。

示例输出

```
Enter the principal: 1500
Enter the rate of interest: 4.3
Enter the number of years: 4

After 4 years at 4.3%, the investment will
be worth $1758.
```

约束

- 提示利率输入的是百分比（像 15，而不是 0.15）。在程序中要除以 100。
- 对于小数部分，不足 1 美分的要向上取整。
- 确保输出格式化为货币形式。

挑战

- 确保本金、利率和年限输入的都是数值，如果输入不合法，程序不会继续处理。

- 修改程序，使用一个名为 `calculateSimpleInterest` 的函数，接受利率、本金和年限，返回这笔投资的到期总额。
- 除了打印到期总额，也打印每年年末的总额。

13 确定复利

只有在想快速猜测一下时才会使用单利。大部分投资都会使用复利公式，它更为精确。该公式需要把指数引入到程序中来。

编写一个计算复利的程序。程序会让用户输入本金、投资年限、利率及每年计算利息的次数。

公式为：

$$A = P(1+\frac{r}{n})^{nt}$$

其中

- ❑ P 为本金
- ❑ r 为年利率
- ❑ t 为投资年限
- ❑ n 为每年计算利息的次数
- ❑ A 为到期总额

示例输出

```
What is the principal amount? 1500
What is the rate? 4.3
What is the number of years? 6
What is the number of times the interest
is compounded per year? 4
$1500 invested at 4.3% for 6 years
compounded 4 times per year is $1938.84.
```

约束

- 提示利率输入的是百分比（像 15，而不是 0.15）。在程序中要除以 100。
- 对于小数部分，不足 1 美分的要向上取整。
- 确保输出格式化为货币形式。

挑战

- 确保所有输入都是数值类型，如果输入不合法，程序不会继续处理。
- 创建一个以相反方式工作的程序，确定要达到某个具体目标，需要投入多少本金。
- 将该程序实现为一个 GUI 应用，当任何值发生变化时，自动更新结果。

本章回顾

作为程序员，我们目前所做的许多事就是选取公式，然后将其转变为代码。你还将编写 invoice、报表、税额计算程序和货币转换器，以及像计算地图上两点之间距离这样复杂的东西。学习编写代码，可不仅是选择已有的公式，然后将其转换为某个算法。

程序员日常还要做很多事情：让计算机对值做比较，并作出相应的响应。进入下一章，处理这些问题吧。

第 4 章

作出决策

到目前为止，我们编写的程序都有点简单。不过有时需要根据用户的输入作出决策。从这里开始，编程就更有挑战了。程序变得越来越长，越来越复杂，而测试也更困难了。测试计划的重要性进一步提升；为确保正确性，必须测试所有可能的解释输入的方式。

那么，如何在程序中作出决策呢？大部分编程语言都有一个 `if` 语句，可以比较两个值。在 JavaScript 中，`if` 语句是这样的：

```
if (userInput === 'Hello') {
    // 做某件事
}
```

如果输入是 `Hello`，花括号之间的代码便会运行。这是一个简单的 `if` 语句。如果输入的是其他信息，则什么都不会发生。有时这正是你想要的。其他情况下你可能想做别的事，这时就可以添加一个 `else` 语句：

```
if (userInput === 'Hello') {
    // 做某件事
} else {
    // 做其他事
}
```

有时可能不仅是从两个中选一个：

```
if (userInput === 'Hello') {
    // 做某件事
} else if (userInput === 'Goodbye') {
    // 做与第一件事不同的事
} else {
    // 做其他事
}
```

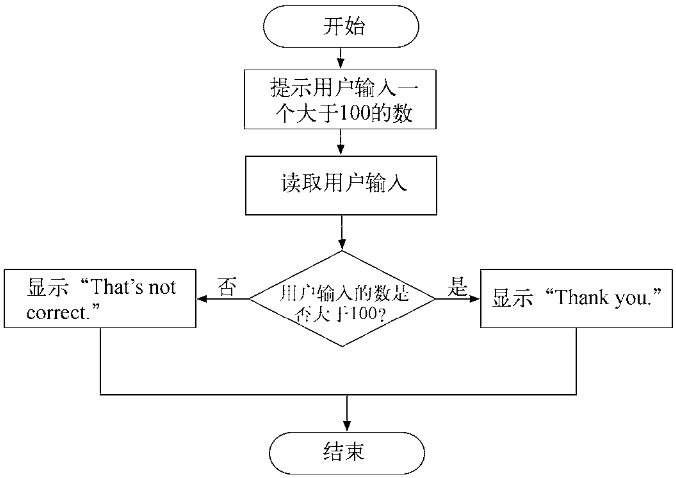
如果有很多可能的结果，那可能正适合 switch 语句大显身手：

```
switch(userInput) {
    case: "Hello"
        // 做某件事
        break;
    case: "Goodbye"
        // 做与第一件事不同的事
        break;
    case: "How was your day?"
        // 做与前两件事不同的事
        break;
    default:
        // 做其他事
}
```

在一个较大的程序中，可能在每一部分或多个阶段都必须做不同的计算。也可以在 if 语句中嵌套 if 语句，你可能会不时这么做。然而，过度使用嵌套 if 语句会使代码很难阅读，甚至随着时间的推移，代码会变得很难维护。因此，当你渐入佳境，你会探索处理决策的不同解决方案。

编写代码只是问题的一小部分。想出编写什么代码更为困难。流程图可以帮你以图形化方式审视要解决的问题，当你认真考虑决策逻辑时，流程图就派上用场了。

比如，编写一个程序，提示用户输入大于 100 的数。如果输入的数大于 100，需要显示 “Thank you.”；如果小于等于 100，则显示 “That's not correct.”。可以创建一个这样的流程图：



这个流程图很容易就能转换成伪代码：

```
Initialize output to ""
Initialize userInput to ""
Prompt for userInput with "Enter a number greater than 100"

IF userInput is greater than 100 THEN
  output = "Thank you."
ELSE
  output = "That's not correct."
END

Display output
```

这种方式可以帮你更好地理解问题，也有助于发现遗漏之处。看一下这里的算法。你看到我遗漏了什么重要的事情了吗？

在比较之前，我忘记将用户的输入从字符串转换成数字了。有些语言会报错，发现这个问题，但其他语言会继续运行，从而导致逻辑错误。画出流程图，写出伪代码，通过这样的方式，我能够将自己的意图快速传达给你，这样，在我花时间编写代码之前，你就能发现我的逻辑中是不是有遗漏的步骤或缺陷。

在做本章的练习时，尝试使用流程图和伪代码来确定程序的算法，然后将其变为代码。

下面将从一些简单的决策着手解决问题，比如“如果该条件发生，则做这件事”。然后你将看到如何解决二选一的问题。然后是更复杂的问题，一个决策的结果会引发另一个决策。这时计划工具就派上用场了。

14 税额计算程序

并不总是需要复杂的控制架构来解决简单的问题。有时候，某个程序只是在一种情况下需要多走一步，而其他情况下什么都不用做。

编写一个简单的程序，计算订单税额。该程序应该提示用户输入订单金额和所在州。如果是威斯康星州（输入“WI”），必须加收 5.5%的税。对于威斯康星州的居民，该程序应该显示订单金额、税额和总金额；对于其他居民，只显示总金额。

示例输出

```
What is the order amount? 10
What is the state? WI
The subtotal is $10.00.
The tax is $0.55.
The total is $10.55.
```

或

```
What is the order amount? 10
What is the state? MN
The total is $10.00
```

约束

- 仅使用一个简单的 `if` 语句实现该程序，不使用 `else` 子句。
- 确保美分部分向上取整。
- 在程序的最后，使用单条输出语句来显示程序的结果。

挑战

- 允许用户以大写、小写或大小写混用的方式输入州的简称。
- 允许用户以大写、小写或大小写混用的方式输入州的全称。

15 密码验证

密码验证是通过比较用户提供的值和之前存储的已知值实现的，以此判断密码正确与否。

编写一个简单程序，验证用户登录凭证。该程序必须提示用户输入用户名和密码。程序会将用户给出的密码与已知密码比较，如果匹配，程序应该显示“Welcome!”，不匹配则显示“I don’t know you.”。

示例输出

```
What is the password? 12345
I don't know you.
```

或

```
What is the password? abc$123
Welcome!
```

约束

- 使用 `if/else` 语句。
- 确保该程序区分大小写。

挑战

- 研究如何阻止在输入时密码以明文形式显示在屏幕上。
- 创建一个用户名和密码的映射，确保用户名和密码组合都匹配。
- 使用跨平台加密工具 Bcrypt 对密码进行编码，将哈希保存在映射中，而不是保存明文密码。在提示用户输入密码时，使用 Bcrypt 加密输入的密码，然后与映射中保存的值比较。

16 法定驾驶年龄

除了测试是否相等，还可能需要看一个数与已知值比较是大还是小，并显示相应的消息。

编写一个程序，让用户输入年龄，将其与法定驾驶年龄 16 比较。如果用户是 16 岁，或者更大，程序应该显示“You are old enough to legally drive.”。如果用户不到 16 岁，则显示 “You are not old enough to legally drive.”。

示例输出

```
What is your age? 15
You are not old enough to legally drive.
```

或

```
What is your age? 35
You are old enough to legally drive.
```

约束

- 使用单条输出语句。
- 使用三元运算符编写该程序。如果所用语言不支持三元运算符，则使用普通的 `if/else` 语句，同时仍然使用单条输出语句来显示消息。

挑战

- 如果用户输入的是小于 0 的数，或者不是数字，显示错误消息，让用户输入一个合法的年龄。

- 不再将法定驾驶年龄硬编码到程序逻辑中，而是研究不同国家的规定，创建一个驾驶年龄和国家的查询表。提示用户输入年龄，显示该用户在哪个国家可以合法驾驶。

17 计算血液中的酒精含量

有时候必须根据某些输入执行更复杂的计算，然后利用结果作出某一决策。

编写一个程序，提示输入体重、性别、喝了几瓶、酒中的酒精量及最后一次饮酒到现在的时间。使用如下公式计算血液中的酒精含量（Blood Alcohol Content，BAC）：

$$\text{BAC} = (A \times 5.14 / W \times r) - 0.015 \times H$$

其中

- A 是饮用的总的酒精量，单位为盎司
- W 是以磅为单位的体重
- r 为酒精分配率：
  - 男性为 0.73
  - 女性为 0.66
- H 是最后一次饮酒到现在的时间

比较 BAC 与 0.08，显示是否可以合法驾驶。

示例输出

```
Your BAC is 0.08
It is not legal for you to drive.
```

约束

- 不允许输入非数字值。

挑战

- 处理公制单位。
- 按州查询法定 BAC 限值，并提示输入州名。根据计算出的 BAC，显示一条消息，指出在该州是否可以合法驾驶。
- 开发一个移动应用，方便记录每次饮酒情况，每次输入饮酒信息，更新 BAC。

18 温度转换程序

我们经常需要确定程序的哪一部分应该根据用户的输入或其他事件来运行。

创建一个程序，将温度从华氏温度（F）转换成摄氏温度（C），或者相反。提示用户输入起始温度。程序应该提示转换类型并执行转换。

转换公式为：

$$C = (F - 32) \times 5 / 9$$

和

$$F = (C \times 9 / 5) + 32$$

示例输出

```
Press C to convert from Fahrenheit to Celsius.
Press F to convert from Celsius to Fahrenheit.
Your choice: C

Please enter the temperature in Fahrenheit: 32
The temperature in Celsius is 0.
```

约束

- 确保支持大小写的 `C` 和 `F`。
- 使用的输出语句要尽可能少，避免输出字符串重复。

挑战

- 修改程序，确保输入为数值。如果不是，则不允许用户继续。

- 将程序分解为执行计算的函数。
- 以 GUI 方式实现该程序，当任何值发生变化时，自动更新结果。
- 修改程序，以支持开氏温度。

19　计算身高体重指数

经常有这样的需求：看看某个值是不是在特定的范围内，并根据结果来修改程序的流程。

创建一个程序，用一个人的身高（以英寸为单位）和体重（以磅为单位）计算其体质指数（Body Mass Index，BMI）。该程序应提示用户输入身高和体重。

使用如下公式计算 BMI：

$$bmi = (weight / (height \times height)) \times 703$$

如果 BMI 在 18.5 和 25 之间，显示这个人为正常体重。如果在该范围之外，则提示用户偏瘦或超重，建议就医咨询。

示例输出

```
Your BMI is 19.5.
You are within the ideal weight range.
```

或

```
Your BMI is 32.5.
You are overweight. You should see your doctor.
```

约束

- 确保程序只接受数值数据。如果数据不合法，则不允许继续。

挑战

- 让程序同时支持英制单位和公制单位。公制单位的公式稍有不同。

- 对于英制单位，提示英尺和英寸，并将英尺转换为英寸，这样就不用用户自己算了。
- 使用 GUI 实现该程序，用滑动条表示身高和体重。当用户拖动滚动条时，即时修改结果。使用不同的颜色和文本来指示健康与否。

20　多州税收计算程序

更复杂的程序中，有些决策可能会嵌套在其他决策中；当一个决策确定之后，其他决策也相应确定。

创建一个税收计算程序，处理多个州以及每个州内不同城市/郡县的税收。该程序应提示用户输入订单金额以及订单履约所在州。

对于威斯康星州居民，提示输入居住城市/郡县。

- ❑ 如果是欧克莱尔（Eau Claire），额外增加 0.005 的税收。
- ❑ 如果是邓恩县（Dunn County），额外增加 0.004 的税收。

伊利诺伊州居民要交 8%的税，城市/郡县不再加税。其他所有州不收税。对于威斯康星州和伊利诺伊州的居民，该程序会显示税收和税后总额，对于其他人，则显示总额。

示例输出

```
What is the order amount? 10
What state do you live in? Wisconsin
The tax is $0.50.
The total is $10.50.
```

约束

- ❑ 确保跟钱相关的数值都向上取整到最接近的美分。
- ❑ 在程序的最后，使用一条输出语句来显示程序结果。

挑战

- 支持你所在的州和城市/郡县。
- 支持用户以大写、小写、混用大小写等形式输入州名缩写和城市/郡县名。
- 支持用户以大写、小写、混用大小写等形式输入州的全名。
- 使用数据结构来实现该程序，避免嵌套 `if` 语句。

21 从数字到名字

很多程序都会以某种形式将信息显示给最终用户，而程序内部则以不同形式表示。比如，在屏幕上可能会显示“Blue”这个词，而在背后会以某个数值或内部值来表示该颜色。这是因为程序可能要支持不同语言的用户，比如西班牙语，显示的文本也会有所不同。

编写一个程序，将从 1 到 12 的数转换成相应的月份。提示用户输入数字，显示对应的英文月份：比如输入 1，显示 January；输入 12，则显示 December。如果输入的值超出该范围，则显示相应的错误信息。

示例输出

```
Please enter the number of the month: 3
The name of the month is March.
```

约束

- 使用 switch/case 语句实现该程序。
- 使用单条输出语句。

挑战

- 使用一个映射（map）或词典（dictionary）实现，去掉 switch 语句。
- 支持多种语言。在程序开始提示用户选择语言。

22 比较数字

将输入和一个已知的值比较，非常普通。更多时候，我们需要处理一组输入。

编写一个程序，让用户输入 3 个数。首先确认所有数字各不相同。如果存在相同的数，退出程序，否则显示其中最大的。

示例输出

```
Enter the first number: 1
Enter the second number: 51
Enter the third number: 2
The largest number is 51.
```

约束

- 手动编写该算法。不要使用寻找列表中最大值的内置函数。

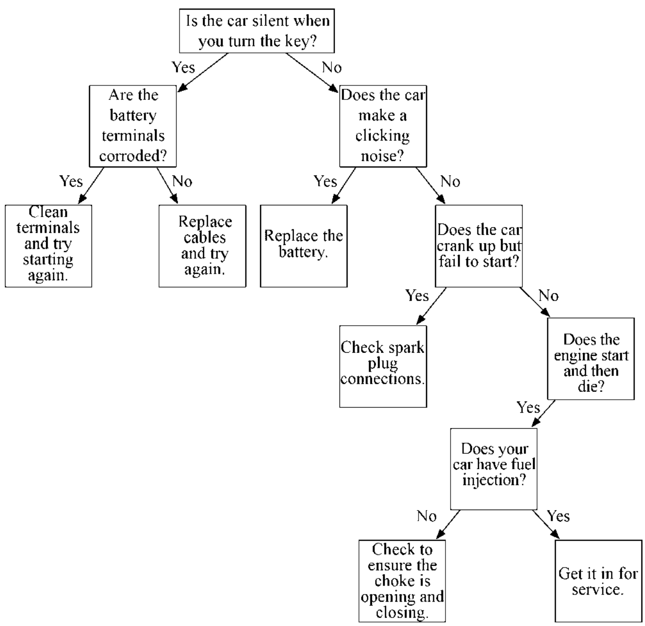
挑战

- 修改程序，记录下用户已经输入的值，防止用户输入已经输入过的值。
- 修改程序，要求输入 10 个数，而不是 3 个。
- 修改程序，不限制输入个数。

23　定位汽车问题

专家系统是一类人工智能程序，它利用一个知识库和一组规则，像人类专家那样执行某项任务。有很多这样的网站，用户通过回答一系列问题来诊断自己的病情。也有很多软硬件公司提供在线故障诊断工具，帮助用户在呼叫人工服务之前解决简单的技术问题。

编写一个程序，通过一系列问题引导用户定位汽车存在的问题。使用如下决策树来构建该系统：



示例输出

```
Is the car silent when you turn the key? y
Are the battery terminals corroded? n
```

```
The battery cables may be damaged.
Replace cables and try again.
```

约束

- 只问与当前状况和前面答案相关的问题。不要一下子让用户输入所有信息。

挑战

- 研究规则引擎和推理引擎。这些是基于规则和事实解决复杂问题的强大手段。为你所用的编程语言确定一个规则引擎，并用该引擎来解决上面的问题。

本章回顾

决策处理是软件开发的关键组成部分。菜单系统就是决策处理驱动的。玩游戏时，当我们按下一个键，屏幕上的角色是跳是跑还是射击，也是通过决策处理来确定的。在真正实现时，也是靠决策处理将晦涩的错误代码转换成人们可以读懂的错误消息。

在学习本章的这些程序时，你可能已经注意到，与之前的程序相比，本章必须做更多测试，因为可能的输出更多了。代码中的分支越多，可能的结果就越多。

在学习下一章之前，请再次检查一下你的程序中的逻辑。是不是覆盖了所有可能结果？比如 BMI 计算程序，如果 BMI 恰好在边界上，会怎么样？使用的比较运算符是不是正确？

如果自信代码没有问题，那就开始学习下一章，将函数引入到我们的程序中。

第 5 章

函　　数

我们的程序越来越复杂了。即使我们尽力将输入、处理和输出分开，随着程序愈加复杂，找起东西来也是越来越困难。

但是我们可以使用函数来组织代码，甚至可以创建可复用的组件。

函数就像主程序内更小的程序。下面的 JavaScript 代码，定义了一个将两个数相加的函数：

```
function addTwoNumbers(firstNumber, secondNumber) {
  return(
    firstNumber + secondNumber
  );
}
```

addTwoNumbers 函数接收两个数作为输入，执行计算，然后将结果返回给程序的其余部分。使用方法如下：

```
var sum = addTwoNumbers(1,2);
console.log(sum);
```

函数的另一个好处是，逻辑被封装在函数体中，可以修改，而不会影响使用它的程序。比如，我们的函数接收两个值，但我们是这样调用的：

```
var sum = addTwoNumbers("1","2");
console.log(sum);
```

程序的输出会是 12，因为 JavaScript 会将字符串连接起来，而不会将其转换为数字。但是我们可以修改 `addTwoNumbers` 函数，自动将输入转换成数字，这样该函数就总能工作了。

我们经常要拿到一个函数的结果，将其发送给另一个函数。有时也会基于一个函数的结果来作出决策。有些编程语言完全基于函数，比如 Elixir 和 Clojure。这些就是所谓的*函数式*编程语言。

在解决本章的问题时，将你的代码组织为函数。尝试将主算法封装到一个函数中，然后在程序的其他部分调用。或者更进一步，为捕获输入和构建输出等操作也创建函数。

本章篇幅不长，这是有意为之，因为当完成这些练习时，你应该再回到前面的程序，看看是不是能用函数改进那些程序的组织。

24 字母易位词检查程序

使用函数将逻辑从其余代码中剥离，使其更容易阅读和维护。

编写一个程序，比较两个字符串是否为字母异位词（anagram），也就是说两个词包含的字母是相同的，只是顺序不同。该程序应提示用户输入两个字符串，输出如下所示。

示例输出

```
Enter two strings and I'll tell you if they
are anagrams:
Enter the first string: note
Enter the second string: tone
"note" and "tone" are anagrams.
```


约束

- 使用一个名为 `isAnagram` 的函数来实现该程序，接收两个词作为参数，返回 `true` 或 `false`。在主程序中调用该函数。
- 检查这两个词的长度是否相同。

挑战

- 在不使用内置语言特性的前提下完成该程序。使用选择或循环逻辑开发自己的算法。

25 检查密码强度

函数可以帮我们抽象掉复杂操作，还可以帮我们构建可复用的组件。

开发一个程序，基于如下规则确定给定密码的强度。

- 如果只包含数字，而且少于 8 个字符，则为非常弱的密码。
- 如果只包含字母，而且少于 8 个字符，则为弱密码。
- 如果包含字母，至少有一个数字，而且不少于 8 个字符，则为强密码。
- 如果包含字母、数字和特殊字符，而且不少于 8 个字符，则为非常强的密码。

示例输出

```
The password '12345' is a very weak password.
The password 'abcdef' is a weak password.
The password 'abc123xyz' is a strong password.
The password '1337h@xor!' is a very strong password.
```

约束

- 创建 `passwordValidator` 函数，以密码为参数，返回一个可以帮我们评估密码强度的值。不要让函数返回字符串，未来可能需要支持多种语言。
- 使用单条输出语句。

挑战

- 开发一个 GUI 应用或 Web 应用，实时以图形和文本形式提供反馈。当用户输入一个密码时，确定其强度并显示结果。

26 计算还清信用卡欠款所需的时间

还清信用卡欠款所需的时间可能比想象的长得多。计算公式并不是那么直观。用一个函数隐藏该公式的复杂性，代码可以组织得更合理。

编写一个程序，确定信用卡欠款要几个月还清。程序应该让用户输入信用卡欠款额、该卡的年利率（Annual Percentage Rate，APR）及每月还款额，然后返回还款所需月数。

计算公式为：

$$n = -\frac{1}{30} \times \frac{\log\left(1 + \frac{b}{p}(1-(1+i)^{30}\right)}{\log(1+i)}$$

其中

- n 为还清所需月数
- i 为日利率（APR 除以 365）
- b 为欠款额
- p 为每月还款额

示例输出

```
What is your balance? 5000
What is the APR on the card (as a percent)? 12
What is the monthly payment you can make? 100

It will take you 70 months to pay off this card.
```

约束

- 提示输入卡的 APR，内部做除法。
- APR 输入的是百分比去掉百分号的数字部分，不是小数。
- 使用名为 `calculateMonthsUntilPaidOff` 的函数，以欠款额、APR 和每月还款额为参数，返回还清欠款所需月数。不要在该函数外使用这些参数值。
- 计算钱的时候，涉及“分”的小数都向上取整。

挑战

- 修改公式，使程序可以接收还款月数，计算每月还款额。创建一个版本,让用户选择是计算还清所需月数,还是计算每月所需还款额。

27 验证输入

较大的函数并不是非常好用，可维护性也不是很好。将程序的逻辑分解为只做一件事的更小的函数，很有意义。程序可以依次调用这些函数来完成工作。

编写一个程序，提示输入名字、姓氏、工号和邮编。根据如下规则确保输入是合法的。

- 必须填写名字。
- 必须填写姓氏。
- 名字和姓氏至少 2 个字符长。
- 工号必须是“AA-1234”这样的格式，即两个字母、一个连字符，后面跟四个数字。
- 邮编必须是数字。

对于不正确的数据，显示相应的错误消息。

示例输出

```
Enter the first name: J
Enter the last name:
Enter the ZIP code: ABCDE
Enter an employee ID:  A12-1234
"J" is not a valid first name. It is too short.
The last name must be filled in.
The ZIP code must be numeric.
A12-1234 is not a valid ID.
```

或

```
Enter the first name: Jimmy
Enter the last name: James
Enter the ZIP code: 55555
Enter an employee ID:  TK-421
There were no errors found.
```

约束

- 为每类验证编写一个函数。然后创建 `validateInput` 函数，它接收所有的输入数据，再调用具体的验证函数。
- 使用单条输出语句来显示输出。

挑战

- 使用正则表达式来验证输入。
- 将该程序实现为 GUI 应用或 Web 应用，当字段失去焦点时，直接给出反馈。
- 如果输入不合法，则让用户重新输入。

本章回顾

现在是时候回头看看前面的章节，再做做那些练习了。定位出程序的主算法，将其封装到一个函数中。看看将一个函数分解为更小的函数是不是有帮助。毕竟没有人说不能从一个函数调用另一个。然后看看那些程序中是不是有重复的函数。如果有，就想办法复用，而不是将代码从一个程序复制到另一个程序。

如果一切就绪，那就开始下一章。下一章中我们会看到一些需要让计算机重复处理过程的问题。

第 6 章

重　　复

如何让计算机反复做同样的事？当然不必把代码敲上很多遍，对不对？

根据结构化程序理论，可以使用三种基本的控制结构来解决计算机程序问题：顺序、选择和循环。顺序方式就是以恰当的顺序一步接一步地处理。选择是基于条件作出决策。前面已经讲过这两种。最开始的程序有大量的顺序处理，然后有些程序要基于条件作出决策，于是我们又接触了选择。

但是如果要重复程序的某些部分，同时想避免复制代码，则要使用循环——指定一组指令在条件成立时重复执行。可以这么考虑，“当用户想输入更多值时，不断要求输入”，或“反复执行这 5 步，直到没有记录了”。

如何重复，取决于我们想要的结果。我们可能想重复特定次数，或者对姓名列表中的每个条目执行一次，或者是直到用户告诉我们结束。

所用的编程语言会影响解决问题的方式。比如，在 Go 语言中，如果想从 5 减到 1，代码可以这么写：

```
counter := 5
for counter <= 1 {
  fmt.Println(counter)
  counter--;
}
```

计数器从 5 开始，打印，然后减 1。代码会一直执行，直到计数器小于等于 1。

C 风格的语言（如 JavaScript）提供了 `for` 循环，支持将计数器变量定义和增量控制作为循环声明的一部分：

```
for(var counter = 5; counter <= 1; counter--) {
  console.log(counter);
}
```

Ruby 开发者可能会选择不同的实现方式。在 Ruby 中，整型数就是支持循环的对象。

```
5.times do |counter|
  # counter is 0-based.
  puts 5 - counter
end
```

但是某些语言，如 Elixir，并没有提供循环，而是依赖递归：

```
defmodule Recursion do
  def loop(n) when n <= 1, do: IO.puts n

  def loop(n) do
    IO.puts n
    loop(n - 1)
  end
end

loop(5)
```

当函数直接或间接地调用到自身时，就出现了递归。在前面的例子中，loop 函数调用了自身，每次减 1，直到 n 小于 1。不是所有的语言都对递归做了优化；在某些语言中，如果某个函数调用自身太多次，栈中会有其多个副本，最终会填满栈，导致程序崩溃。所以，编程语言的选择又一次决定了能够使用的方法。

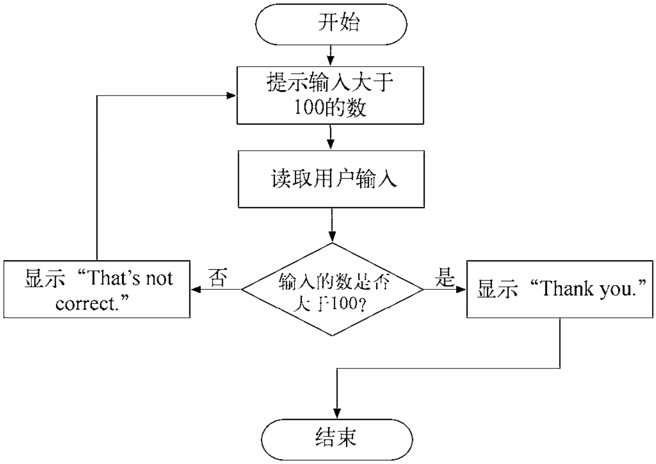
这些例子都是计数型循环，往上或往下数到某个已知的值。但有些时候，可能需要不同的条件来停止循环，比如当输入特定值时停止。

```
var value;
var keepGoing = true;
while(keepGoing) {
  value = prompt("Enter a number or 0 to stop.");
  keepGoing = value !== 0;
  // 更多处理
}
```

或者是当文件中有更多行要处理时继续，或者是当数据库中有更多记录要显示时继续。

本章的练习要求你使用重复来高效地解决它们。在阅读每个程序的问题描述时，仔细思考一下，重复的次数是固定的，还是不固定的。然后选择最合适的方法。

你会发现，使用流程图将有助于确认程序的逻辑。还记得第 4 章中使用流程图的例子吗？我们必须提示用户输入大于 100 的数。如果输入不符合要求，则终止程序。不过我们可以使用重复不断提问，下面使用流程图来表示该逻辑。



流程图帮我们看清了程序包含着一个重复性过程；当用户输入了不正确的值时，再次提示他们。由此，我们就可以确定使用代码实现该过程的最佳方式了。

本章的练习有点简单，不过它们可以帮助你为后续严重依赖重复的章节做好准备。依次做做这些练习题，为后面的学习打下坚实的基础吧。

28 数字相加

在前面的程序中，我们写了多条输入语句，让用户重复输入。但是，用循环来处理重复输入问题更为高效。

编写一个程序，提示用户输入 5 个数字，计算它们的和。

示例输出

```
Enter a number: 1
Enter a number: 2
Enter a number: 3
Enter a number: 4
Enter a number: 5
The total is 15.
```

约束

- 提示必须使用重复，比如计数型循环，而不是使用 3 个独立的提示。
- 在编写程序之前画一个流程图。

挑战

- 修改程序，让用户输入想求和的数字的个数，而不是硬编码。确保在比较之前将输入转化为数字。
- 修改程序，只对数字值求和，对于非数字值，默默拒绝。无效的输入也计算在输入次数内；换句话说，如果要求和的数字是 5 个，那就提示 5 次。

29 处理错误的输入

回报率 72 规则（rule of 72）是快速评估资产翻番所需时间的一个方法，就是用 72 除以预期回报率，得到预期年数。它可以帮我们计算股票、债券或存款是否合适自己。因为计算机不能计算除以 0，所以对于编写测试并防止错误的输入而言，该程序也有很好的意义。当用户键入无效的输入值时，不要退出程序，而是不断提示用户输入，直到获得有效的输入值。

编写一个快速计算程序，提示输入某项投资的回报率，计算资产翻番需要多少年。

公式为：

$$年数 = 72/r$$

其中 r 为设定回报率。

示例输出

```
What is the rate of return? 0
Sorry. That's not a valid input.
What is the rate of return? ABC
Sorry. That's not a valid input.
What is the rate of return? 4
It will take 18 years to double your initial investment.
```

约束

- 不允许用户输入 0。
- 不允许输入非数字值。

- ❑ 使用循环来捕获错误输入，以确保用户输入有效值。

挑战

- ❑ 当用户输入 0 时，显示一条不一样的错误信息。

30 乘法表

创建一个程序，生成从 0 到 12 的乘法表。

示例输出

```
0 X 0 = 0
0 X 1 = 0
....
12 x 11 = 132
12 x 12 = 144
```

约束

- ❑ 使用一个嵌套循环来完成该程序。

挑战

- ❑ 创建一个图形界面程序。使用下拉列表来修改基数。当用户选中数字时，生成或更新乘法表。
- ❑ 生成一个下面这样的表。

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12
2	0	2	4	6	8	10	12	14	16	18	20	22	24
3	0	3	6	9	12	15	18	21	24	27	30	33	36
4	0	4	8	12	16	20	24	28	32	36	40	44	48
5	0	5	10	15	20	25	30	35	40	45	50	55	60
6	0	6	12	18	24	30	36	42	48	54	60	66	72
7	0	7	14	21	28	35	42	49	56	63	70	77	84
8	0	8	16	24	32	40	48	56	64	72	80	88	96
9	0	9	18	27	36	45	54	63	72	81	90	99	108
10	0	10	20	30	40	50	60	70	80	90	100	110	120
11	0	11	22	33	44	55	66	77	88	99	110	121	132
12	0	12	24	36	48	60	72	84	96	108	120	132	144

31　卡蒙内心率

在循环中，可以自己控制计数器增加多少，未必是每次加 1。

考虑开发一款健身程序，为防止运动过度，我们想计算目标心率。卡蒙内（Karvonen）心率公式就是一个可以用来确定心率的方法。开发一个程序，提示输入年龄和安静时的心率。使用卡蒙内公式，按运动强度从 55%到 95%，计算目标心率。如示例输出所示，以表格形式生成结果。公式为：

$$TargetHeartRate = (((220 - age) - restingHR) \times intensity) + restingHR$$

示例输出

```
Resting Pulse: 65  Age: 22

Intensity    | Rate
-------------|--------
55%          | 138 bpm
60%          | 145 bpm
65%          | 151 bpm
 :           : (其他行省略)
85%          | 178 bpm
90%          | 185 bpm
95%          | 191 bpm
```

约束

- 不要硬编码百分比。使用一个循环，控制百分比从 55 增加到 95。
- 确保心率和年龄输入的是数字。如果输入不合法，不允许用户继续。
- 以表格形式显示结果。

挑战

- 以 GUI 方式实现该程序，支持用户用滑动条来控制运动强度，随着滑动条的移动实时更新界面。

32 猜数字游戏

编写一个猜数字程序，难度有三级。从第 1 级到第 3 级，数字分别位于 1 ~ 10、1 ~ 100 和 1 ~ 1000。

首先提示玩家选择难度，然后启动游戏。计算机会在相应区间内生成一个随机数，让玩家来猜。每当玩家做出猜测之后，给玩家一个线索，提示输入的数字是大了还是小了。计算机应记录所猜次数。一旦玩家猜对，则显示玩家所猜次数，并询问玩家是否要继续玩。

示例输出

```
Let's play Guess the Number.
Pick a difficulty level (1, 2, or 3): 1
I have my number. What's your guess? 1
Too low.  Guess again: 5
Too high. Guess again: 2
You got it in 3 guesses!
Play again? n
Goodbye!
```

约束

- 不允许输入任何非数字的值。
- 在游戏过程中，将非数字数据视作猜错。

挑战

- 将所猜次数对应如下评语：

  猜 1 次：“You’re a mind reader!”

猜 2~4 次："Most impressive."

猜 3~6 次："You can do better than that."

猜 7 次或以上："Better luck next time."

- 记录之前猜过的数字，如果用户再次输入，则警告已经猜过这个数。这也算一次错误的猜测。
- 使用数字栅格，以图形化方式实现该游戏。当一个数字被点击过之后，将其从屏幕中去掉。

本章回顾

学到这里，你应该对自己的能力非常自信了。你已经掌握了条件逻辑，知道如何使用函数，现在也知道了如何让程序的某些部分重复执行。你甚至可以捕获错误输入。本书前面的很多程序，都可以从你学的这些知识中获益，所以在继续学习后面的章节之前，可以修改其中一些程序。或许可以从防止某些计算类程序的无效输入入手。

如果没有首先理解数组之类的数据结构，则很难应对很多现实中的问题，所以本章篇幅很短。下一章就会讲数据结构了，我们开始吧！

第 7 章

数据结构

只有最简单的程序能够侥幸靠将数据保存在变量中解决。但是下面这些程序会让你思考将数据保存在列表或名值对中，甚至是组合使用它们。

取决于所选的编程语言，你可能会寻找数组（array）、列表（list）、哈希（hash）、哈希映射（hashmap）、词典（dictionary）、关联数组（associative array）或映射（map）。尽管不同语言中名字可能有所不同，但概念是相似的。

我们使用数据结构将数据组织到一起。简单起见，我将使用数组和映射这两个术语。

数组是保存一系列值的数据结构：

```
colors = ["Red", "Green", "Blue"];
```

数组中元素的顺序信息通常会保留下来。可以通过索引得到一个值，索引就是元素在数组中的位置。在大多数语言中，第一个元素的索引是 0：

```
colors = ["Red", "Green", "Blue"];
console.log(colors[0]);
>> "Red"
```

映射是一个将键（key）映射到值（value）的数据结构，用户使用键而非位置来检索数据：

```
person = {name: "Homer", age: 42};
console.log person["name"];
```

数据结构最常见的一种使用情形，是表示来自数据库的一组记录。每条记录用一个映射表示，每个字段是该映射中的一个名值对。记录的集合使用数组表示。

这是用 JavaScript 写的一个例子：

```
var people = [
  {name: "Homer", age: 42},
  {name: "Barney", age: 41}
]
```

下面是用 Elixir 实现的同样的结构：

```
people = [
  %{name: "Homer", age: 42},
  %{name: "Barney", age: 41}
]
```

尽管语法不同，但概念是一样的；使用数据结构保存类似的数据，以便在程序中使用。

数据结构往往会和循环一起使用。比如，如果有一组名字，想打印出其中的每一个，可以迭代该集合，处理这些值。下面是用 JavaScript 实现的一个使用 `for` 循环的例子：

```
var names = ["Ted", "Barney", "Carl", "Tracy"];
for(var i = 0, length = names.length; i < length; i++) {
  console.log(names[i]);
}
```

不过，很多语言还提供了一种替代方案。在 Ruby 中，可以编写像这样的代码：

```
names = ["Ted", "Barney", "Carl", "Tracy"]
names.each { |name| puts name }
```

Elixir 提供了一种类似的方案：

```
names = ["Ted", "Barney", "Carl", "Tracy"]
names
  |> Enum.each fn(name) -> IO.puts name end
```

Java、C#、JavaScript 及其他很多语言都提供了大量用于迭代、排序和操纵列表及其他数据结构的特性。所以在学习本章的过程中，要看一下如何使用这些概念，并利用所学的知识尝试解决本章的问题。

33 神奇 8 号球

数组很适合保存程序中多个可能的响应信息。如果结合随机数生成器，可以根据随机数从数组中随机选一个，这很适合游戏。

创建一个“神奇 8 号球”游戏，提示用户输入一个问题，然后显示“Yes”“No”“Maybe”或“Ask again later”。

示例输出

```
What's your question?  Will I be rich and famous?
Ask again later.
```

约束

- 值应该使用某个伪随机数生成器随机选择。将可能的选择保存在一个列表中，随机选择一个。

挑战

- 将其实现为一个 GUI 应用。
- 如果有的话，使用原生设备库，“摇动”8 号球。

34 从员工列表中删除元素

有时必须基于某个标准从列表中定位并删除一个元素。可能是一副牌，需要丢掉其中一张，丢掉的这张牌就不能再用了。还可能是需要从一组合法的输入中将用过的值删掉。将这些值保存在数组中，处理可以更快些。取决于所选的编程语言，你或许会发现，与修改现有数组相比，创建一个新的数组可能更安全、更高效。

创建一个包含一系列员工姓名的程序。在程序第一次运行时，打印这些姓名。提示用户输入一个员工的姓名，从姓名列表中将其删除。然后显示其余员工的信息，每个员工的信息占一行。

示例输出

```
There are 5 employees:
John Smith
Jackie Jackson
Chris Jones
Amanda Cullen
Jeremy Goodwin

Enter an employee name to remove: Chris Jones

There are 4 employees:
John Smith
Jackie Jackson
Amanda Cullen
Jeremy Goodwin
```

约束

- 使用数组或列表来保存姓名。

挑战

- 如果没找到用户输入的名字，打印一条错误消息，指出这个名字不存在。
- 从一个文件中读入员工列表，该文件中每个员工占一行。
- 将输出写到读入时所用的文件中。

35 选择优胜者

数组并不一定要提前写好。可以获得用户输入，将其保存在一个数组中，然后处理。

创建一个程序，选出某场竞赛或某个抽奖活动的优胜者。提示用户输入选手的名字，直到用户输入空行，然后随机选择一个优胜者。

示例输出

```
Enter a name: Homer
Enter a name: Bart
Enter a name: Maggie
Enter a name: Lisa
Enter a name: Moe
Enter a name:
The winner is... Maggie.
```

约束

- 使用一个循环获得用户输入，并保存到数组中。
- 使用一个随机数生成器从数组中选出一个值。
- 不要将空行保存到数组中。
- 有些语言要求提前定义数组的长度。你可能需要寻找另一种数据结构，比如 `ArrayList`。

挑战

- 当选择出一个优胜者时，将其从选手列表中删除，支持选出更多优胜者。

- 开发一个 GUI 程序，在选出优胜者之前，先打乱数组中名字的顺序，然后将该数组显示在屏幕上。
- 创建一个单独的竞赛登记应用。使用该程序把所有注册用户拉取过来，然后选出一个优胜者。

36　计算统计信息

在我们这个领域，统计非常重要。当测量响应时间或渲染时间时，收集数据很有意义，这样就能轻松发现异常情况。例如，标准差可以帮我们确认哪些值是离群值，哪些值在正常范围之内。

编写一个程序，提示输入某个网站的响应时间，以毫秒表示。不断让用户输入值，直到用户输入“done”。

该程序应打印平均时间（平均值，mean）、最小时间（min）、最大时间（max）和标准差（standard deviation）。

要计算平均时间（平均值），应：

(1) 计算所有值之和；
(2) 除以值的个数。

要计算标准差，应：

(1) 计算每个值与平均值的差，并计算差的平方；
(2) 计算这些平方值的平均值；
(3) 计算平均值的平方根。

示例输出

```
Enter a number: 100
Enter a number: 200
Enter a number: 1000
Enter a number: 300
Enter a number: done
Numbers: 100, 200, 1000, 300
```

```
The average is 400.
The minimum is 100.
The maximum is 1000.
The standard deviation is 400.25.
```

约束

- 使用循环和数组来执行输入和数学运算。
- 务必不要将“done”放到数组中。
- 务必正确地将数值转换为字符串。
- 保持输入、处理和输出的分离。

挑战

- 使用名为 `mean`、`max`、`min` 和 `standardDeviation` 的函数，参数为数值数组，返回计算结果。
- 让程序从某个外部文件中读入数值，而不是在命令行提示用户输入值。

37 密码生成器

生成一个满足特殊要求的密码，这是计算机可以做的。利用这个例子，可以练习如何结合使用一组已知的值与随机数生成器。

创建一个生成安全的密码的程序。提示用户输入密码的最小长度、特殊字符的个数，以及数字的个数。然后使用那些输入为用户生成一个密码。

示例输出

```
What's the minimum length? 8
How many special characters? 2
How many numbers? 2
Your password is
aurn2$1s#
```

约束

- ❑ 使用列表来保存将用于生成密码的字符。
- ❑ 向密码生成过程加入一些随机性。

挑战

- ❑ 随机将元音字母转换为数字，比如 E 转换为 3，A 转换为 4。
- ❑ 让程序提供一组选项，而不是一个结果。
- ❑ 生成之后，将密码放到用户的剪贴板中。

38 过滤值

有时候需要过滤一下收集到的输入。借助数据结构和循环，处理可以更简单。

创建一个程序，提示输入一组数值，用空格间隔。让程序打印一个只包含偶数的新列表。

示例输出

```
Enter a list of numbers, separated by spaces:  1 2 3 4 5 6 7 8
The even numbers are 2 4 6 8.
```

约束

- 将输入转到一个数组中。在很多语言中，都可以借助某个内置函数将基于特定分隔符的字符串分开，从而轻松地将字符串转成数组。
- 编写自己的算法，不要依赖语言内置的过滤器或类似的枚举特性。
- 使用一个名为 `filterEvenNumbers` 的函数包装该逻辑。该函数接受老数组，并返回新数组。

挑战

- 不再是提示输入数值，而是从任意文本文件中读入行，打印标号为偶数的行。

39 排序记录

在看结果的时候，可能会想对它们做个排序，这样就能快速找到想要的东西，或是快速做一些看得出来的比较。

给定如下数据集：

First Name	Last Name	Position	Separation date
John	Johnson	Manager	2016-12-31
Tou	Xiong	Software Engineer	2016-10-05
Michaela	Michaelson	District Manager	2015-12-19
Jake	Jacobson	Programmer	
Jacquelyn	Jackson	DBA	
Sally	Weber	Web Developer	2015-12-18

编写一个程序，按姓氏（last name）排列所有员工，然后以表格形式打印到屏幕上。

示例输出

```
Name                | Position          | Separation Date
--------------------|-------------------|----------------
Jacquelyn Jackson   | DBA               |
Jake Jacobson       | Programmer        |
John Johnson        | Manager           | 2016-12-31
Michaela Michaelson | District Manager  | 2015-12-19
Sally Weber         | Web Developer     | 2015-12-18
Tou Xiong           | Software Engineer | 2016-10-05
```

约束

- 使用映射的列表表示该数据。

挑战

- 提示用户输入排序规则。支持按合同到期日期、职位或姓氏排序。
- 使用某款数据库（如 MySQL）或键值存储（如 Redis）来保存员工记录。从数据存储中检索回记录。

40 过滤记录

排序记录很有用，不过有时候还需要对结果加以过滤，仅找出或显示我们想要的。

给定如下数据集：

First Name	Last Name	Position	Separation date
John	Johnson	Manager	2016-12-31
Tou	Xiong	Software Engineer	2016-10-05
Michaela	Michaelson	District Manager	2015-12-19
Jake	Jacobson	Programmer	
Jacquelyn	Jackson	DBA	
Sally	Weber	Web Developer	2015-12-18

编写一个程序，让用户输入一个搜索字符串，找到所有名字或姓氏中包含该字符串的记录。

示例输出

```
Enter a search string:  Jac

Results:
Name                 | Position           | Separation Date
---------------------|--------------------|----------------
Jacquelyn Jackson    | DBA                |
Jake Jacobson        | Programmer         |
```

约束

- 使用映射的数组或关联数组表示数据。

挑战

- 让搜索区分大小写。
- 加入按职位搜索这一选项。
- 加入一个选项，找出离职日期是 6 个月前或更早的所有员工。
- 从文件中读入数据。

本章回顾

数据结构可以帮我们结构化组织数据。你会发现数组和映射无处不在。当使用数据库时，记录会以数组形式返回，我们会循环处理。当要读取或修改配置文件时，也会使用数组和映射。我都记不清有多少次被要求拿一个数组然后以某种方式排序了。你也会的。

列表和映射是不错的开始，不过你也可以定义自己的数据结构，比如 `ShoppingCart`。

到目前为止，我们是从用户输入中获得数据，或是自己在代码中写好。下一章，我们将从文件中读入数据。

第 8 章

使用文件

到目前为止，我们处理过的所有程序都是从终端用户那里获得输入，或者是使用硬编码的值。但是很多程序使用文件来存储数据。你用的操作系统及其程序会不断地将日志写入文件，你访问的网站也是如此。还有很多应用使用文件来存放配置数据。游戏也是使用文件来存储玩家到达某个检查点时的存档数据。

就拿编程语言来说，比如你用来实现本书练习题的那门语言，也要使用文件。你把源代码输入到某个文件中，编译器或者解释器会将你写的内容转换为计算机可以运行的程序。

本章的练习会要求你处理文件和文件夹，所以你需要研究一下在自己使用的编程语言中应该怎么做。有些语言内置了读文件的特性，而有些语言没有，不过一般可以使用库来处理。

你也需要研究一下不同的方式。你可能会发现，如果一行一行地处理文件，或者将其看作一个数据流，程序可能会执行得更快。有些文件太大了，无法一下子全部加载；而有些情况可能会要求你在处理之前先读入整个文件。

请注意，如果你是在浏览器中使用 JavaScript，因为浏览器会阻止读写本地文件系统，所以无法在不加修改的情况下做这些练习题。不过可以使用 Node.js。

41 姓名排序程序

要想比较舒服地处理程序中的某个文件，按字母顺序或某种方式对其中的内容做一下排序，是个不错的选择。

创建一个程序，读入下列姓名列表。

```
Ling, Mai
Johnson, Jim
Zarnecki, Sabrina
Jones, Chris
Jones, Aaron
Swift, Geoffrey
Xiong, Fong
```

按字母顺序排序该列表，然后如示例输出所示，将排好序的列表打印到一个文件中。

示例输出

```
Total of 7 names
-----------------
Ling, Mai
Johnson, Jim
Jones, Aaron
Jones, Chris
Swift, Geoffrey
Xiong, Fong
Zarnecki, Sabrina
```

约束

- 不要将姓名的个数硬编码到程序中。

挑战

- ❑ 重新实现程序，从用户那里读入姓名，每次读一个，将排序后的结果打印到一个文件中。
- ❑ 使用该程序处理一个比较大的数据集，看看表现如何。
- ❑ 用某门函数式编程语言实现该程序，并和之前的实现比较一下。

42 解析数据文件

有时数据是以某种结构化的格式提供的，在处理之前，必须加以分解，并转换成记录。CSV，或者以逗号分隔的值，就是常见的结构化格式。

编写一个程序，读入以下数据文件：

```
Ling,Mai,55900
Johnson,Jim,56500
Jones,Aaron,46000
Jones,Chris,34500
Swift,Geoffrey,14200
Xiong,Fong,65000
Zarnecki,Sabrina,51500
```

处理该记录，并以格式化的表格形式显示结果，间隔均匀，如示例输出所示。

示例输出

```
Last  First  Salary
-------------------------
Ling  Mai 55900
Johnson  Jim 56500
Jones Aaron  46000
Jones Chris  34500
Swift Geoffrey  14200
Xiong Fong65000
Zarnecki Sabrina51500
```

约束

- ❑ 自己编写代码来解析数据，不要使用某种 CSV 解析器。
- ❑ 使用空格实现列对齐。

- 每一列要比该列中最长的值多一个空格。

挑战

- 将工资格式化为带有美元符号（$）和逗号（,）的货币形式。
- 按工资从高到低排序结果。
- 使用某种 CSV 解析库重写该程序，并比较结果。

43　网站生成器

编程语言也可以创建文件和文件夹。

编写一个程序，按如下说明生成一个网站的骨架。

- 提示输入网站名称。
- 提示输入网站作者。
- 询问用户是否想要一个保存 JavaScript 文件的文件夹。
- 询问用户是否想要一个保存 CSS 文件的文件夹。
- 生成一个 index.html 文件，网站的名字包含在`<title>`标签下，作者包含在`<meta>`标签下。

示例输出

```
Site name: awesomeco
Author: Max Power
Do you want a folder for JavaScript? y
Do you want a folder for CSS? y
Created ./awesomeco
Created ./awesomeco/index.html
Created ./awesomeco/js/
Created ./awesomeco/css/
```

挑战

- 在 Windows、OS X 和 Linux 上，用某种脚本语言实现该程序。
- 将其实现为一个 Web 应用，以 zip 文件格式提供指定的网站。

44 产品搜索

将数据从文件拉取到某个复杂的数据结构中，解析起来会更简单。很多编程语言都支持 JSON 这种流行的数据表示格式。

编写一个程序，以产品名（name）为输入，查找该产品当前的价格（price）和数量（quantity）。产品数据保存在 JSON 格式的数据文件中，看上去是这样的：

```
{
  "products" : [
 {"name": "Widget", "price": 25.00, "quantity": 5 },
 {"name": "Thing", "price": 15.00, "quantity": 5 },
 {"name": "Doodad", "price": 5.00, "quantity": 10 }
  ]
}
```

如果找到该产品，打印产品名，以及对应的价格和数量。如果没找到，则提示没有该产品，并重新开始。

示例输出

```
What is the product name? iPad
Sorry, that product was not found in our inventory.
What is the product name? Widget
Name: Widget
Price: $25.00
Quantity on hand: 5
```

约束

- 文件是 JSON 格式的。使用某种 JSON 解析器，拉取出文件中的值。
- 如果没发现记录，再次提示。

挑战

- 确保产品搜索区分大小写。
- 如果某个产品没有找到，询问用户是否应该添加该产品。如果是，询问价格和数量，并将其保存到 JSON 文件中。确保新添加的产品可以立即搜索到，而不用重启程序。

45 单词查找

有时候需要读入一个文件，加以修改，然后将修改过的版本写到一个新文件中。

给定一个输入文件，读取该文件，找到所有出现单词“utilize”的位置，然后将其替换成“use”。将修改过的文件保存到一个新文件中。

示例输出

对于给定的输入文件：

```
One should never utilize the word "utilize" in
writing. Use "use" instead.
```

程序应该生成：

```
One should never use the word "use" in
writing. Use "use" instead.
```

约束

- ❑ 提示用户提供输出文件的名字。
- ❑ 将输出写到一个新文件中。

挑战

- ❑ 修改程序，记录做了几次替换，当程序结束时，将该结果打印到屏幕上。
- ❑ 创建一个配置文件，将单词“bad”映射为“good”，使用该文件来代替硬编码的值。
- ❑ 修改该程序，以支持处理文件夹，而不仅仅是处理单个文件。

46 词频统计

如果要创建单词云或进行其他类型的单词分析，了解一个单词在一句或一段话中出现的频率会很有帮助。如果是在大量文字上运行，那就更有用了。

开发一个程序，读入一个文件，统计文件中单词出现的频率。然后构造一个直方图，显示单词及其出现频率，并将直方图显示在屏幕上。

示例输出

给定文本文件 words.txt，内容如下：

```
badger badger badger badger mushroom mushroom
snake badger badger badger
```

该程序会生成如下输出：

```
badger:   *******
mushroom: **
snake:    *
```

约束

- 按单词出现的频率从高到低排列。

挑战

- 实现图形用户界面程序，生成柱状图。
- 提供一个非常大的输入文件，比如莎士比亚的《麦克白》[1]，测试

① http://shakespeare.mit.edu/macbeth/full.html

程序的计算性能。修改算法，让单词统计的速度尽可能快。

- 用另一种语言编写该程序，比较两个实现的处理时间。

本章回顾

理解如何用你所选的编程语言读、写和操纵文件，这是一项关键任务，现在你也做了不少练习了。当然，更多的练习有助于磨练技能，所以可以考虑再看一下第 7 章中的问题，加以修改，以支持从文件读取记录，而不是从内存中获得。

但是在这个相互联系日益紧密的世界中，我们可能必须与互联网上的其他服务交互。下一章我们来学习一下。

第 9 章

使用外部服务

会使用提供数据的外部服务，这是程序员具备的最重要的技能之一。Twitter、Flickr、Facebook 和 Google 等网站都会通过 API（Application Programming Interface，应用编程接口）暴露其数据。

应用向 API 发出请求，API 会提供数据作为响应，然后我们在应用中处理数据。数据可能是 XML 或 JSON 格式的，不过有时候可能必须自己满屏幕找结果。

有些 API 是可以免费获得的，而有些则需要注册为开发者才有使用权。这就给我们的程序带来了额外的复杂性，因为我们要想办法安全地存储密钥（key）。职业软件开发者通常会使用像 Git 这样的版本控制软件，如果密钥保存在源代码中，一不小心就会被上传到版本控制系统中，或者更糟糕，被上传到像 GitHub 这样的网站上，也就是公开了。这样的事儿出了可不是一次两次了。

如果是在浏览器中使用 JavaScript，可不能直接把密钥信息放到 JavaScript 代码中，因为运行该程序的所有人都能查看全部源代码，并窃取到密钥。所以应该考虑使用自己的服务器端代理来处理请求。

要完成本章的练习，需要了解第三方 API 的工作原理，以及如何将其集成到自己的程序中。需要阅读每个 API 的文档，理解如何获得数据，数据是什么格式的，然后看看如何从自己的程序中请求数据，以及如何处理结果。

47 谁在太空中?

想不想准确知道现在谁在太空中？Open Notify API 提供了该信息。访问 http://api.open-notify.org/astros.json，不但能看到目前太空中有多少人，还能看到他们的名字，以及他们在哪个航天器上。

创建一个程序，获得该数据，并将从 API 得到的数据显示在表格中。[①]

示例输出

```
There are 3 people in space right now:

Name                | Craft
--------------------|------
Gennady Padalka     | ISS
Mikhail Kornienko   | ISS
Scott Kelly         | ISS
```

约束

- 每次程序运行时，直接从 API 读取数据并解析结果。不要将数据以文本格式下载后再读入解析。

挑战

- 确保表头的宽度能满足相应列中最长的值。
- 不要重复航天器的名字，按照航天器来对人员进行分组。
- 能不能可靠地将结果按姓氏的字母顺序排序？要细心，有些人的名字中有空格，比如“Mary Sue Van Pelt”。

① ISS，International Space Station，国际空间站。注意，因为测试时间不同，读者获得的结果可能和书中不同。——译者注

48 抓取天气

今天室外天气不错吧？气温如何？是不是该拿上外套？

使用 OpenWeatherMap API（http://openweathermap.org/current），创建一个程序，提示用户输入城市名，然后返回该城市当前的气温。

示例输出

```
Where are you? Chicago IL
Chicago weather:
65 degrees Fahrenheit
```

约束

- 将程序中处理天气 feed 信息的部分与显示结果的部分分开。

挑战

- 该 API 会给出日出、日落时间，湿度，以及对天气的一个描述。以一种有意义的方式显示这些数据。
- 该 API 会给出以度表示的风向。将其转换为东、西、南、北、东南、西北之类的词。
- 开发一种模式，让天气程序告诉我们今天是什么天儿。比如，如果气温超过 70 华氏度，万里无云，就说今天适合外出。
- 同时以摄氏度和华氏度显示温度。
- 基于相关信息，确定需要带外套还是带伞。

49 Flickr 照片搜索

有些服务提供了搜索功能，而且我们可以对要获得的结果加上很多控制。我们要做的就是构造正确的请求。

创建一个带图形用户界面的程序，让用户输入搜索字符串，然后显示与其匹配的照片。使用 Flickr 的公开照片 feed（https://www.flickr.com/services/feeds/docs/photos_public/）。

示例输出

程序应该像下面这样显示照片：



约束

- 因为这是一个图形用户界面程序，所以需要显示从 API 获得的照片。如果用的是 JavaScript，请使用 HTML 和 DOM，不要使用 jQuery 或任何外部框架。如果用的是 Java，尝试使用 Swing 构建一个桌面应用，或者开发一个 Android 应用。如果所用的语言没有富 GUI 工具包，可以创建一个 HTML 页面，用本机的浏览器打开。

挑战

- 如果用的是 JavaScript，使用 Angular、Ember 或 React 实现程序。如果觉得应付得来，可以挨个试试。
- 使用 Twitter API 找到与搜索字符串有关的推文，显示在照片旁边。

50　电影推荐

外部服务提供的数据为我们创建自己的程序提供了一个很好的起点。

编写一个程序，显示给定电影的信息。提示用户输入查询信息，查找相关电影，如果有的话，显示电影名、上映年代、分级、片长和故事梗概等。如果观众评分超过 80%，推荐用户马上观看。如果评分低于 50%，建议用户无论如何都不要看了。

示例输出

```
Enter the name of a movie: Guardians of the Galaxy

Title: Guardians of the Galaxy
Year: 2014
Rating: PG-13
Running Time: 121 minutes

Description: From Marvel...

You should watch this movie right now!
```

约束

- 使用 Rotten Tomatoes API（http://developer.rottentomatoes.com/），并获取一个 API 密钥。

挑战

- 开发一个图形版本的程序，以图形化方式显示电影海报以及分级信息。
- 研究如何缓存收集到的电影数据，这样就不用一直调用外部 API 了。提供一个设置缓存过期的方法。

51 向 Firebase 提交笔记

有些外部服务允许更新数据，而不仅仅是读取。Firebase[①]就是一个这样的服务，允许用户创建自己的数据库，用于为 Web、移动和桌面应用保存数据。得益于其基于 JSON 的 API，我们可以使用任何编程语言。

创建一个简单的命令行应用，支持保存和显示笔记，使用 Firebase 保存笔记。该应用应该支持如下命令：

- ❑ `mynotes new Learn how to invert binary trees` 会保存这条笔记；
- ❑ 使用 `mynotes show` 可以显示已有的所有笔记。

示例输出

```
$ mynotes new Learn how to invert binary trees
Your note was saved.

$ mynotes show
2050-12-31 - Learn how to invert binary trees
2050-12-30 - Notetaking on the command line is cool.
```

约束

- ❑ 创建一个保存 API 密钥的配置文件。
- ❑ 使用 REST 文档（https://www.firebase.com/docs/rest/）来代替预先创建的客户端库。

① https://www.firebase.com/

挑战

- 创建一个更通用的应用，支持搜索和查看笔记。
- 用一个客户端库替换自己的实现。
- 添加为笔记打标签的功能。
- 回头看一下第 8 章的问题，修改为使用 Firebase。

52 创建自己的时间服务

用外部服务是一回事，能够创建供他人使用的服务也很重要，这样就可以支持想使用我们所提供服务的开发者了。

创建一个简单的 Web 服务，以 JSON 格式返回当前时间，如：`{"currentTime": "2050-01-24 15:06:26" }`。

然后创建一个客户端应用，连接到前面创建的 Web 服务，解析响应，显示时间。

示例输出

```
The current time is 15:06:26 UTC January 4 2050.
```

约束

- 在服务器应用中，确保在发送响应时将内容类型（content type）设置为 `application/json`。
- 用尽可能少的代码构建服务器应用。

挑战

- 构建一个新的服务器，随机显示一条名言。将名言保存在数组中，随机选择一个显示。
- 选择与服务器端所用编程语言不同的语言，编写一个显示名言的客户端组件。

本章回顾

现代的程序通常会依赖第三方服务，所以知道如何使用它们是非常有帮助的，不过你很有可能会发现，在自己的工作中也会用到这种模式。用平台原生语言编写的移动应用，向用某种服务器端语言编写的中央后端读写数据，这很常见。Web 应用通常使用客户端 JavaScript 来配合服务器端 JSON API。在本章练习中学到的经验是很有用的。

是时候处理一些更为健壮的程序，把在本书前面所学的所有知识融会贯通了。

第 10 章

完整的程序

如果已经完成了本书中的其他练习，你可能想找找更大的挑战，强化自己的编程技能。本章的练习需要你把前面学到的所有知识汇总到一起。有些练习要求你走出舒适区，可能必须研究一下所用编程语言的标准库，才能解决这里的某些问题。

在处理这些练习时，思考一下在本书开始时所探索的问题解决过程。看看问题陈述，想想如何将问题分解为更小的单元。考虑如何编写一些测试计划，以确保自己知道程序是正确的。

此外，看看能不能找出某些在之前的程序中做过的模式或事情。使用同样的方法解决不同的问题，这是很常见的。

53 待完成事项清单

我们就从一个入门级的程序开始，即“待完成事项清单”（Todo List），虽然并不新鲜，但是作为例子还是很不错的。编写一个命令行的待完成事项清单程序，需要满足下列要求。

- 提示用户输入一项家务或任务。将任务保存到持久性位置，这样当程序重启时，任务还存在。
- 允许用户输入尽可能多的任务，当用户输入空任务时，停止接受输入。不要保存空任务。
- 显示所有任务。
- 允许用户移除一项任务，表示该任务已经完成。

约束

- 将数据保存在某个外部的数据源中。
- 如果使用的是服务器端语言，考虑将数据持久化到 Redis 中。
- 考虑将数据持久化到像 Parse 或 Firebase 这样的第三方服务中。

挑战

- 仅使用前端技术，在 Web 浏览器中实现该程序。研究使用 IndexedDB 来保存任务项。
- 将前端实现为 Android 或 iPhone App，将其连接到使用服务器端语言编写的后端。创建用于检索任务清单、创建新任务和将任务标记为完成的 API。

54 短网址服务

编写一个 Web 应用，类似于 https://goo.gl/，将用户输入的长 URL 转换为短 URL。

- 有一个接受长 URL 的表单。
- 生成一个短的、特有的 URL，如/abc1234；并将短 URL 和长 URL 一起保存在持久性数据存储中。
- 当访问短 URL 时，程序将访问者重定向到长 URL。
- 记录短 URL 被访问的次数。
- 有一个短 URL 统计页面，如/abc1234/stats。当访问该 URL 时，显示短 URL、长 URL 以及短 URL 的访问次数。

约束

- 该应用必须使用可以公开访问的持久性数据存储。这意味着本地内存系统并不合适。
- 表单中不允许输入无效的 URL。

挑战

- 检查重复的 URL。如果已经存在，则不再创建新的短 URL。
- 使用 Redis 作为数据存储。
- 使用 RavenDB 作为数据存储。
- 记录每个短 URL 被访问的日期和时间，使用图形库，以曲线图表示请求。

55 文本分享

创建一个 Web 应用，支持用户分享一段文本，类似于 http://pastie.org。编写的程序应该满足下面这些要求。

- 用户将文本输入一个文本区域，并保存文本。
- 文本应该保存到数据存储中。
- 程序应该生成一个 URL，可用于检索已保存的文本。
- 当某个用户访问该 URL 时，程序显示相应文本，并邀请该用户编辑该文本。
- 当某个用户点击编辑按钮时，复制文本并显示在同一界面下，以创建新的文本片段。

约束

- 使用主键以外的信息构造 URL，比如生成别名（slug）。研究一下 SHA 或 MD5 哈希。

挑战

- 修改程序，使每条粘贴内容支持 Markdown 格式。
- 修改程序，使编辑功能可以编辑现有节点，同时可以记录之前的版本。
- 实现一个 API，使命令行、原生或移动应用都可以添加新的文本片段或查看片段。

56 记录财产

编写一个程序，记录个人财产。该程序允许用户输入物品、序列号和估算价格。程序应该能以 HTML 和 CSV 格式打印表格状报表，像这样：

Name	Serial Number	Value
Xbox One	AXB124AXY	$399.00
Samsung TV	S40AZBDE4	$599.99

约束

- 将数据以 JSON、XML 或 YAML 格式保存在一个持久性本地文件中。
- 对于每种物品的价格，需要输入数字值。

挑战

- 修改程序，支持保存照片。如果正在为移动设备开发应用，允许用户用摄像头拍照。
- 支持搜索物品。

57 多选琐事问答应用

创建一个多选琐事问答应用。

- 从一个文件中读入问题、答案和干扰选项。
- 当玩家开始游戏时
  - 随机选择问题；
  - 以随机顺序显示答案和干扰选项；
  - 记录正确答案的序号；
  - 如果玩家选择错误，结束游戏。

约束

- 使用文件数据库或本地数据文件，而非键值存储或关系型数据库。

挑战

- 为问题添加一个难度字段，随着游戏的进行，给出的问题难度越来越大。
- 扩展该程序，增加一个模式，允许父母或老师添加、编辑或删除问题与答案。

下一步干什么？

借助这些程序，但愿你对所选的编程语言有了一定程度的掌握，可以开始思考想要解决的自己的问题了。要深入挖掘一门语言或框架，最好的办法是用它解决自己的痛点。想想生活中想要解决的问题，或者

试试重写某个现有应用。编写自己的卡路里计算应用、番茄钟定时器或购物清单应用。

学习软件开发行业的其他重要工具。探索测试驱动开发，使用所用语言中的工具编写单元测试和验收测试。研究一下用 Git 做版本控制，将代码提交到 GitHub[1]上，让其他人也能看到。或者是应用自己的新技能为开源项目作出贡献。这对于学习他人的经验和发展自己的职业都是非常好的。

当学习下一门语言时，可以再拿起这本书，从头开始，不过要结合新的思考方式来解决类似的问题。编码愉快！

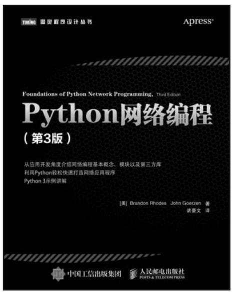
① http://github.com

延 展 阅 读



- Amazon编程入门类榜首图书
- 从基本概念到完整项目开发，帮助零基础读者迅速掌握Python编程
- 上到有编程基础的程序员，下到10岁少年，想入门Python并达到可以开发实际项目的水平，本书是最佳选择！

书号：978-7-115-42802-8
定价：89.00 元



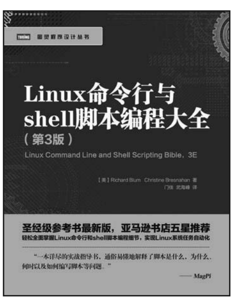
- 从应用开发角度介绍网络编程基本概念、模块以及第三方库
- 利用Python轻松快速打造网络应用程序
- Python 3示例讲解

书号：978-7-115-43350-3
定价：79.00 元



- Amazon榜首畅销书全新升级，Android入门进阶不二之选
- 全面覆盖Android开发知识点，通过实战项目手把手教你逐步写Android应用

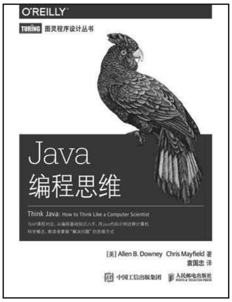
书号：978-7-115-42246-0
定价：109.00 元



- 圣经级参考书最新版，亚马逊书店五星推荐
- 轻松全面掌握Linux命令行和shell脚本编程细节，实现Linux系统任务自动化

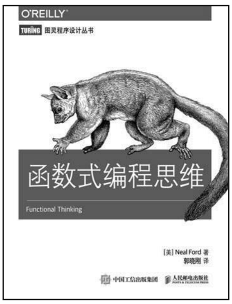
书号：978-7-115-42967-4
定价：109.00 元

延 展 阅 读



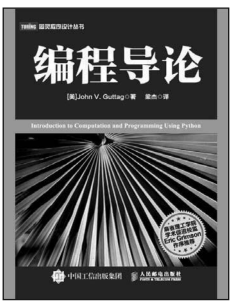
- 与AP课程对应，从编程基础知识入手，用Java代码示例诠释计算机科学概念，教读者掌握“解决问题”的思维方式
- 具备Java编程思想，学会像计算机科学家一样思考

书号：978-7-115-44015-0
定价：59.00 元



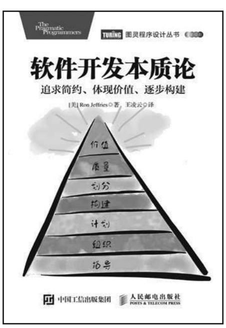
- 大量函数式编程示例，Scala、Clojure、Groovy、Java 8实现
- 彻底搞懂函数式编程思维，并了解在函数式语境下的设计模式和代码重用
- ThoughtWorks知名软件架构师Neal Ford作品，Agile Developer创始人Venkat Subramaniam推荐

书号：978-7-115-40041-3
定价：49.00 元



- 麻省理工学院开放式课程最受欢迎的计算机课程的教材
- Python语言特性和编程方法贯穿全书，帮助读者在学习Python 的同时学会用计算来解决有趣问题

书号：978-7-115-38801-8
定价：59.00 元



- 敏捷先驱为你直观呈现软件开发简约之道，实践极限编程
- 构建高质量软件系统必读

书号：978-7-115-44110-2
定价：39.00 元

“如果你正想学习一门新的编程语言，那你也应该选择这本书。你将从中学到如何从基本原理出发解决问题，为后面的学习和工作打下坚实的基础。我学到了很多，希望你也大有收获。”

——Stephen Orr，Impact Applications高级软件工程师

“实践是学习新编程语言的最佳方法，而这本书就是这方面的绝佳资源。因为这本书是语言无关的，所以也有无限的重读价值。在诸多技术图书中，这种特质极为少见。”

——Jason Pike，theswiftlearner.com软件工程师

“对于任何一个想学习一门全新语言的人而言，这本书都是非常棒的。不管是新程序员还是老程序员，都能从这本书的练习题中获益良多。初学者可以舒服地学习这本书，有经验的程序员也能看到很多挑战。”

——Alex Henry，JAMF Software测试工程师

Exercises for Programmers

57 Challenges to Develop Your Coding Skills

学习并掌握一门编程语言的最佳方式是用它去解决问题，而本书正是为想要动手实践的程序员设计的。书中基于日常软件开发中经常遇到的实际问题提炼了57道练习题，以帮助程序员磨练技艺、提升技能。这些练习题由浅入深，首先从简单的程序入手，然后逐步过渡到解决更难的问题。如果你是一名新手，这些练习题可以帮你打开编程的大门。如果你是一位经验丰富的程序员，也可以运用这些练习题来快速地学习一种新语言或新的编程风格。

The Pragmatic Programmers

图灵社区：iTuring.cn

热线：(010)51095186转600

分类建议 计算机/程序设计

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-44680-0

9 787115 446800 >

ISBN 978-7-115-44680-0

定价：39.00元

看完了

如果您对本书内容有疑问，可发邮件至 contact@turingbook.com，会有编辑或作译者协助答疑。也可访问图灵社区，参与本书讨论。

如果是有关电子书的建议或问题，请联系专用客服邮箱：ebook@turingbook.com。

在这可以找到我们：

微博 @图灵教育：好书、活动每日播报
微博 @图灵社区：电子书和好文章的消息
微博 @图灵新知：图灵教育的科普小组
微信 图灵访谈：ituring_interview，讲述码农精彩人生
微信 图灵教育：turingbooks